Alexis Balayre

Future Position Prediction for Pressure Refuelling Port of Commercial Aircraft

School of Aerospace, Transport and Manufacturing
Computational and Software Techniques in Engineering

MSc
Academic Year: 2023–2024

Supervisors: Dr Boyu Kuang and Dr Stuart Barnes
May 2024

School of Aerospace, Transport and Manufacturing
Computational and Software Techniques in Engineering

MSc

Academic Year: 2023–2024

Alexis Balayre

Future Position Prediction for Pressure Refuelling Port of Commercial Aircraft

Supervisors: Dr Boyu Kuang and Dr Stuart Barnes
May 2024

This thesis is submitted in partial fulfilment of the requirements for the degree of MSc.

Academic Integrity Declaration

I declare that:

- the thesis submitted has been written by me alone.
- the thesis submitted has not been previously submitted to this university or any other.
- that all content, including primary and/or secondary data, is true to the best of my knowledge.
- that all quotations and references have been duly acknowledged according to the requirements of academic research.

I understand that to knowingly submit work in violation of the above statement will be considered by examiners as academic misconduct.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ML	Machine Learning
DL	Deep Learning
AI	Artificial Intelligence
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Unit
EKF	Extended Kalman Filter
AAGR	Autonomous Aircraft Ground Refueling
AGR	Aircraft Ground Refueling
UAV	Unmanned Aerial Vehicle
AAR	Autonomous Aerial Refueling
DGPS	Differential Global Positioning System
SVM	Support Vector Machine
HOG	Histogram of Oriented Gradients
SOTA	State-of-the-Art
AIS	Automatic Identification System
GPS	Global Positioning System
IoU	Intersection over Union
TP	True Positive
FP	False Positive
FN	False Negative
TN	True Negative
AP	Average Precision
AR	Average Recall
SSD	Single Shot Multibox Detector
YOLO	You Only Look Once
IoT	Internet of Things
AFRL	Air Force Research Laboratory
AR3P	Autonomous & Robotic Remote Refueling Point
BIPRS	Brightness Invariant Port Recognition System

Chapter 1

Introduction

1.1 Background and Motivation

Ground pressure refuelling is a standard method used to refuel commercial aircraft safely and efficiently. This process involves using a system of underground fuel pipelines and hydrants at aircraft parking spots [5]. When an aircraft is ready for refueling, a hydrant dispenser vehicle connects to the hydrant pit and delivers fuel to the aircraft through a flexible hose [39] (see Figure 1.1). This method allows for high fuel flow rates and significantly reduces aircraft turnaround times [5]. However, this process also presents several challenges, particularly in terms of safety and accuracy. In the past, there have been several incidents involving ground pressure refueling, including fuel spills, overfills, and equipment failures [34, 12]. Fortunately, with the advancement of technology, many of these challenges will be addressed, and the refueling process will become safer and more efficient.

Figure 1.1: Pressure Refuelling of a Commercial Aircraft. Photo Credit: Tom Boon/Simple Flying [3]

The aviation industry is undergoing a significant transformation with the development of the airport of the future, commonly known as 'Smart Airport' or, more recently, 'Airport 4.0'. The concept of Smart Airport encompasses the use of cutting-edge information technologies, such as the Internet of Things (IoT), Artificial Intelligence (AI), and Blockchain, to monitor, analyse, and integrate real-time data on the airport's status. This integration aims to achieve optimal operational efficiency and enhance the quality of service. Taking the concept a step further, Airport 4.0 envisages an airport driven entirely by AI, capable of making autonomous decisions thanks to self-learning mechanisms. This advance aims to automatically predict and

manage various airport scenarios, making it easier to automate numerous processes. The result is a substantial reduction in operating costs and error rates [31].

Among these, automated refuelling systems play a crucial role in ensuring efficient and accurate refuelling of aircraft. However, one of the main challenges of this automation process is the accurate detection of the aircraft's refuelling port, which is relatively small and can easily be obscured by other visual elements on or near an aircraft. For example in the Figure 1.2, the refuelling port is located on the wing of the aircraft and can be difficult to detect due to motion blur, occlusion, or being out of view. This challenge is further compounded by the fact that aircraft refuelling ports can vary in size, shape, and location depending on the aircraft type and manufacturer. Scanning the entire area of each video frame is both time-consuming and inaccurate. It is therefore essential to develop a more efficient and accurate method of locating the refuelling port.

(a) Motion Blur Example (b) Occlusion Example (c) Out-of-View Example

Figure 1.2: Challenges in Detecting Aircraft Refuelling Port

1.2 Research Gap

Despite significant advancements in autonomous aircraft ground refuelling technologies, critical challenges remain, particularly in the accurate detection and positioning of the refuelling port. The small size and varied locations of refuelling ports, often obscured by visual elements like motion blur and occlusions, complicate this task. Existing systems have made progress using machine vision, but they are limited by inefficiencies in scanning entire video frames and inaccuracies under different environmental conditions. Furthermore, while current methodologies leveraging convolutional neural networks (CNNs) and Kalman filters have improved detection accuracy, they still struggle with real-time performance and adaptability in dynamic environments. The robustness of these systems in varied lighting conditions and their capability to handle different refuelling port types and obstructions need enhancement. Additionally, the application of advanced deep learning models like Long Short-Term Memory (LSTM) networks, Recurrent Neural Networks (RNNs), and Transformers in this context is underexplored.

1.3 Aim and Objectives

This thesis aims to address the previous research gaps by developing a framework for predicting the future position of commercial aircraft refuelling ports using advanced Object Detection models and Deep Learning to leverage the spatial-temporal relationship between frames in a video.

1. Conduct a comprehensive review of state-of-the-art methods for Object Detection, Object Tracking, and Deep Learning Sequence Models.

2. Annotate and preprocess video datasets of aircraft refuelling ports to ensure high-quality training and testing data.

3. Design and develop a framework for accurately tracking and predicting the future position of aircraft refuelling ports.

1.4 Technological Contributions

This thesis makes several technological contributions aimed at advancing the field of automated aircraft refuelling systems. The primary contribution is the integration of advanced deep learning models, including Long Short-Term Memory (LSTM) networks and Gated Recurrent Unit (GRU) networks, to predict the future positions of refuelling ports by leveraging their superior spatio-temporal data processing capabilities. Another significant contribution is the development of an innovative framework that combines object detection, target tracking, and future position prediction to enhance the accuracy and efficiency of automated refuelling systems. This framework is designed to handle the small pixel ratio of refuelling ports efficiently, thus optimising the detection and tracking process. The use of Extended Kalman Filtering (EKF) further refines predictions, ensuring real-time adaptability and accuracy, which is crucial for practical applications in busy airport environments.

1.5 Thesis Layout

The following sections of this thesis provide a detailed exploration and analysis of the methodologies, experiments, and findings related to the development of an advanced framework for predicting the future positions of aircraft refuelling ports. The Chapter 2 (Literature Review) presents an overview of the current state-of-the-art methods in automated aircraft refuelling systems, object detection, and deep learning for spatio-temporal prediction. The Chapter 3 (Methodology) describes the step-by-step approach taken in dataset preparation, framework design, and model training. The Chapter 4 (Experiment Design) outlines the experimental setups and comparison studies conducted to evaluate the performance of the proposed models. In the Chapter 5 (Results and Discussion), the outcomes of these experiments are presented and analysed, offering insights into the effectiveness and implications of the research findings. Finally, the Chapter 6 (Conclusion and Future Work) summarises the key contributions of the thesis, reflects on the significance of the results, and proposes directions for future research to further advance the field of automated aircraft refuelling systems.

Chapter 2

Literature Review

2.1 Automated Refulling Systems in the Aviation Industry

Ground refueling operations are essential to maintaining aircraft availability and operational efficiency. The transition from manual to automated systems is designed to improve the safety, efficiency and reliability of these operations [35]. The concept of Autonomous Aircraft Ground Refueling (AAGR) emerged in the 1980s in the United States to address the US Air Force's need to protect ground personnel from potential threats during refueling operations [40]. In the early 1990s, Bennett et al. [4] introduced the Brightness Invariant Port Recognition System (BIPRS), marking a significant advancement in machine vision systems for identifying aircraft refuelling ports. In 2010, the Air Force Research Laboratory (AFRL) showcased the world's first Automated Aircraft Ground Refuelling system prototype through a video demonstration. This system featured a robot equipped with a fuel nozzle and a single-point refuelling adapter, enabling autonomous engagement with the aircraft's refuelling panel, as illustrated in Figure 2.1 [8]. This project will give birth to the Autonomous & Robotic Remote Refuelling Point (AR3P) project.

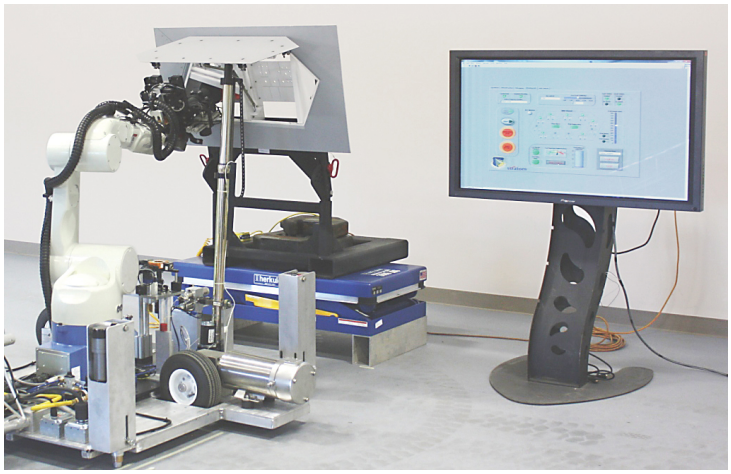


Figure 2.1: AFRL's Automated Aircraft Ground Refueling system prototype robot (Photo Credit: AFRL/RXQ Robotics Group)

The Autonomous & Robotic Remote Refuelling Point (AR3P) project, developed by the U.S. Army, represents a pioneering initiative in unmanned refuelling operations for rotary-wing aircraft. This project leverages advanced robotics, including self-aligning mechanisms

and articulated arms equipped with sensors, to facilitate rapid and safe refuelling processes on non-contiguous battlefields. The AR3P system minimises the time aircraft spend on the ground and enhances safety by reducing soldier exposure at fueling stations. Initially demonstrated in a Limited Initial Capabilities event, the AR3P aims to meet the evolving range and endurance requirements of Army Aviation. The project integrates existing technologies with novel systems designed in-house, supported by commercial off-the-shelf components and additive manufacturing. Currently, AR3P is progressing through its development phases, addressing technical risks, and preparing for further testing and eventual deployment, as shown in Figure 2.2 [15].

Figure 2.2: AR3P Concept Development Prototype Robot (Photo Credit: U.S. Army)

The AR3P project exemplifies the intersection of advanced robotics and practical military applications, highlighting the potential of automated systems to transform operational paradigms. Figure 2.3 provides visual insights into the capabilities of the AR3P system, by performing autonomous hot refuelling. During this test, the robot is equipped with a LIDAR sensor and a camera to detect the aircraft's refuelling port. In Figure 2.3a, the AR3P robot is seen approaching a detected aircraft, demonstrating its autonomous navigation and alignment capabilities. In Figure 2.3b, the AR3P robot is shown engaging the aircraft refuelling port, emphasising its precision and functionality in connecting to the aircraft's fuel port. These images illustrate the practical implementation of robotic technologies in enhancing the safety, efficiency, and speed of refuelling operations, particularly in challenging and hazardous environments [2].

(a) AR3P Robot Approaching Detected Aircraft (Photo Credit: Stratom)

(b) AR3P Robot Engaging Aircraft Refueling Port (Photo Credit: Stratom)

Figure 2.3: AR3P Robot Hot Refueling Demonstration for S-70 Helicopter

Unfortunately, there is very little literature on existing AAGR systems, as most research is carried out by the military and is classified. The most recent papers cover Autonomous Aerial Refueling (AAR) systems, which are used to refuel unmanned aerial vehicles (UAVs) in mid-air. These systems are designed to extend the flight time and range of UAVs by enabling them to refuel without landing. AAR systems are particularly challenging due to the high speeds and altitudes involved, as well as the need for precise measurement and tracking of the relative position between the receiver aircraft and the tanker aircraft are critical, particularly during the docking phase (see figure 2.4) [20, 45]. Zhong et al. [54] propose a robust solution that utilises monocular vision combined with an extended Kalman filter (EKF) to address this challenge. By implementing EKF, the system can provide reliable position estimations and track the drogue within a specified region of interest (ROI), even in the presence of disturbances such as air turbulence. As shown in figure 2.5, this system initialises the state and covariance matrices, predicts the drogue's position, updates the state based on new measurements, and continuously refines the ROI for subsequent image processing. This approach significantly reduces the processing time and improves the detection frequency from 10 Hz to up to 30 Hz by focusing computational resources on the predicted ROI.

Figure 2.4: Autonomous Aerial Refueling (AAR) of X-47B Unmanned Combat Air System Demonstrator (Photo Credit: U.S. Navy)

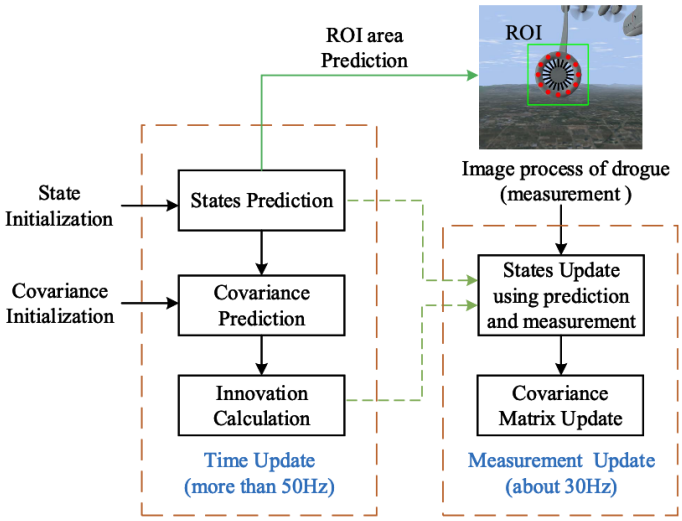


Figure 2.5: Autonomous Air Refueling Detection System with EKF. Source: Zhong et al. [54]

Recent advancements in autonomous ground refuelling have been driven by improvements in computer vision and robotics. Yildirim et al. [50] presented the PosEst system, which combines 2D RGB images with 3D point cloud data to enhance detection accuracy. This system uses a custom-trained EfficientNet-B0 CNN for object detection and leverages the Kalman filter for stable 3D pose estimation (see Figure 2.6). The PosEst method employs a dual approach of high-precision detection and robust tracking. By predicting and updating the object's state in real-time, the Kalman filter facilitates continuous and precise alignment of the fuel nozzle with the refuelling adaptor, even in dynamic environments. This approach significantly reduces the risks associated with manual refuelling and improves operational efficiency and safety.

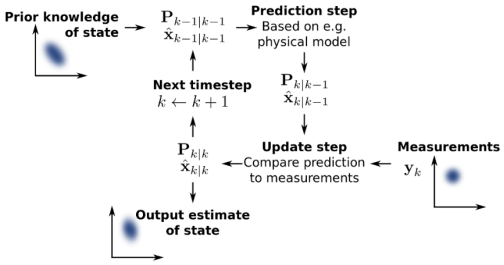


Figure 2.6: Kalman Filter Workflow for Pose Estimation in Autonomous Ground Refueling. Source: Yildirim et al. [50]

One of the primary challenges in AAGR is the accurate detection and positioning of the refuelling port under varying environmental conditions. Robust datasets for scene recognition and machine learning applications have been developed to address these challenges. Kuang et al. [26] introduced a comprehensive dataset for AAGR, addressing significant challenges such as variant illumination conditions, different refuelling port types, and environmental obstructions. The dataset comprises over 26,000 labeled images collected through image crawling from 13 different databases, followed by augmentation to ensure diversity (see Figure 2.7). Additionally, recent innovations have introduced hybrid datasets combining real and synthetic data for training and validating systems [51]. This approach offers a wide range of scenarios and conditions, improving the robustness and accuracy of automated refuelling systems. The development of high-quality datasets is pivotal in improving the robustness and reliability of AAGR systems.

Figure 2.7: AAGR Dataset Overview. Source: Kuang et al. [26]

2.2 Object Detection and Tracking in Computer Vision

In Computer Vision, Object Detection refers to the identification and location of individual objects within an image, providing both spatial information (bounding boxes) and confidence scores, which represent the probability that each detected object belongs to the predicted class [14]. For example, in the following image, there are five detections, including one 'ball' with a confidence level of 98% and four 'people' with confidence levels of 98%, 95%, 97% and 97%.

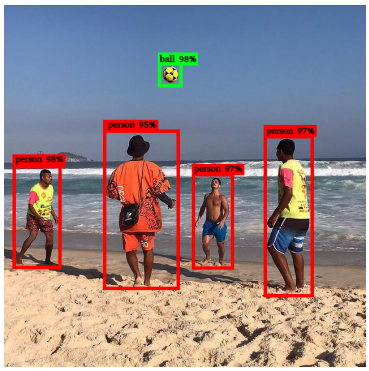


Figure 2.8: Example of outputs from an object detector [14].

Over the last few decades, Object Detection models based on Deep Learning have enjoyed remarkable success. These models fall into two main categories: two-stage detectors and single-stage detectors. On the one hand, two-stage detectors, such as R-CNN [19], Fast R-CNN [18], Faster R-CNN [38] and R-FCN [13], first generate region proposals and then refine these proposals into precise anchor boxes. While these models excel in detection accuracy, they typically suffer from large model sizes and slower detection speeds [24, 49]. On the other hand, single-stage detectors, including the SSD (Single Shot Multibox Detector) [33], YOLO (You Only Look Once) series [37, 36, 6, 10, 16, 22, 29, 23, 44, 47, 43, 48, 42], and RetinaNet [30] directly predict object locations and categories in a single network pass. These models are known for their high detection speeds but sometimes compromise accuracy [24, 49].

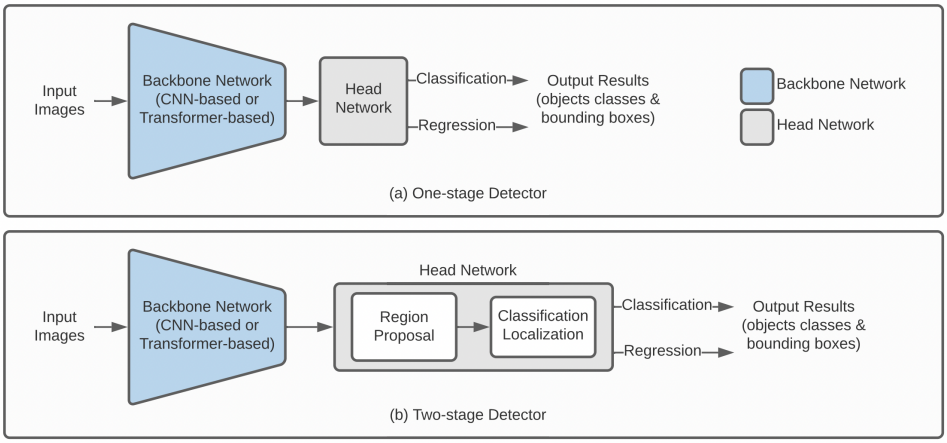


Figure 2.9: Basic deep learning-based one-stage vs two-stage object detection model architectures [24].

YOLOv10, the latest iteration in the YOLO series, marks a significant leap forward in real-time object detection with the introduction of NMS-free training. Traditionally, single-stage detectors relied on Non-Maximum Suppression (NMS) during post-processing to eliminate redundant predictions, as illustrated in Figure 2.10. However, this process can sometimes be overly aggressive, risking the loss of valuable predictions or failing to remove all duplicates effectively, which also adds to computational costs during both training and inference [17].

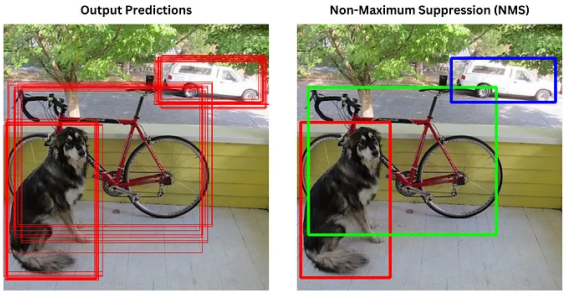


Figure 2.10: Non-Maximum Suppression (NMS) in Object Detection [17].

YOLOv10 overcomes these limitations by implementing a consistent dual assignments strategy that facilitates NMS-free training. This strategy leverages both one-to-many and one-to-one label assignments during training, providing the model with rich supervision while enabling efficient end-to-end deployment. During inference, the one-to-one assignment head is employed, which eliminates the need for NMS and significantly reduces inference time. As depicted in Figure 2.11, the one-to-many head assigns multiple labels to each anchor box, enriching the model's supervision, while the one-to-one head refines these predictions by assigning a single label to each anchor box, ensuring precise and efficient detection [42].

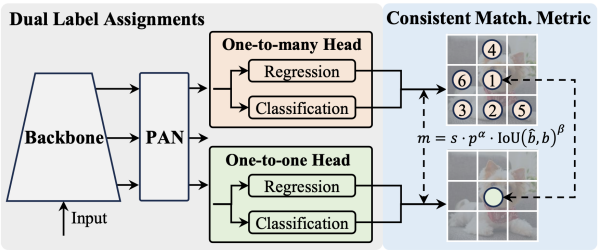


Figure 2.11: YOLOv10 Model Workflow [42]

In addition, YOLOv10 adopts a holistic efficiency-accuracy driven design that optimises the model's architecture across various dimensions. The classification head has been redesigned to be more lightweight, reducing computational overhead while maintaining accuracy. The spatial-channel decoupled downsampling technique separates spatial reduction and channel increase operations, which lowers computational costs and reduces the parameter count. Furthermore, the rank-guided block design minimises redundancy within the model by adapting the complexity of different stages based on their intrinsic rank values, ensuring optimal capacity-efficiency trade-offs. YOLOv10 also introduces large-kernel (see figure 2.12) convolutions selectively in the deeper stages of the network to increase the receptive field, allowing the model to capture more contextual information without a significant increase in computational cost. Moreover, YOLOv10 incorporates a partial self-attention (PSA) mechanism, which integrates the benefits of global context modeling while maintaining a lightweight architecture.

This careful balance of efficiency and accuracy enables YOLOv10 to deliver state-of-the-art performance [42, 17].

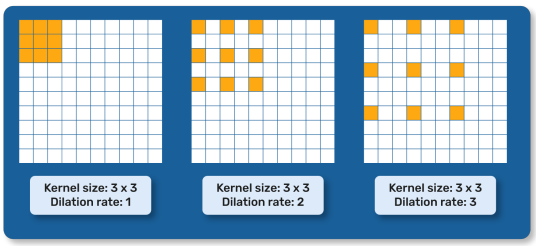


Figure 2.12: Large-Kernel Convolution in YOLOv10 [17]

As shown in table 2.1, extensive testing on standard benchmarks, such as COCO, demonstrates YOLOv10's superior performance in both speed and accuracy. For example, YOLOv10-S is 1.8 times faster than RT-DETR-R18 while also using fewer parameters. Similarly, YOLOv10-B achieves a 46% reduction in latency compared to YOLOv9-C without compromising performance. These results underscore YOLOv10's effectiveness as a real-time end-to-end object detection model, making it well-suited for applications requiring both high speed and accuracy.

Table 2.1: Performance Comparison of YOLO Models with State-of-the-Art Techniques. Latency is reported using official pre-trained models. [f]Latency refers to the forward pass duration without including post-processing. † indicates YOLOv10 results obtained with the original one-to-many training and Non-Maximum Suppression (NMS) [42].

Model	**Params (M)**	**FLOPs (G)**	**AP_{val} (%)**	**Latency[c] (ms)**	**Latency[f] (ms)**
YOLOv6-3.0-N	4.7	11.4	37.0	2.69	1.76
Gold-YOLO-N	5.6	12.1	39.6	2.92	1.82
YOLOv8-N	3.2	8.7	37.3	6.16	1.77
YOLOv10-N	**2.3**	**6.7**	**38.5 / 39.5†**	**1.84**	**1.79**
YOLOv6-3.0-S	18.5	45.3	44.3	3.42	2.35
Gold-YOLO-S	21.5	46.0	45.4	3.82	2.73
YOLO-MS-XS	4.5	17.4	43.4	8.23	2.80
YOLO-MS-S	8.1	31.2	46.2	10.12	4.83
YOLOv8-S	11.2	28.6	44.9	7.07	2.33
YOLOv9-S	7.1	26.4	46.7	-	-
RT-DETR-R18	20.0	60.0	46.5	4.58	4.49
YOLOv10-S	**7.2**	**21.6**	**46.3 / 46.8†**	**2.49**	**2.39**
YOLOv6-3.0-M	34.9	85.8	49.1	5.63	4.56
Gold-YOLO-M	41.3	87.5	49.8	6.38	5.45
YOLO-MS	22.2	80.2	51.0	12.41	7.30
YOLOv8-M	25.9	78.9	50.6	9.50	5.09
YOLOv9-M	20.0	76.3	51.1	-	-
RT-DETR-R34	31.0	92.0	48.9	6.32	6.21
RT-DETR-R50m	36.0	100.0	51.3	6.90	6.84
YOLOv10-M	**15.4**	**59.1**	**51.1 / 51.3†**	**4.74**	**4.63**

Evaluating Object Detection models involves several key metrics to measure their performance. One common metric is Intersection over Union (IoU), which measures the overlap between a predicted bounding box and a ground-truth bounding box, as shown in Figure 2.13.

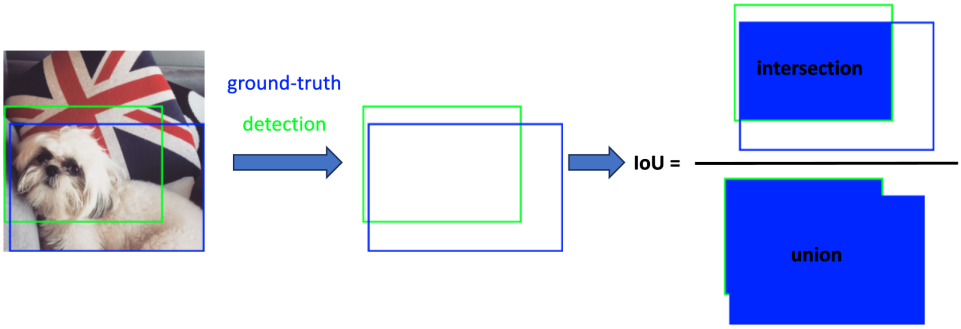


Figure 2.13: Intersection over Union (IoU) between a detection (in green) and ground-truth (in blue). [14]

Based on the IoU metric, a detection can be classified as a **True Positive (TP)** or a **False Positive (FP)** depending on whether the IoU value exceeds a certain threshold (T_{IoU}). If the IoU is above the threshold, the detection is considered correct (TP); otherwise, it is classified as a False Positive (FP). Additionally, **False Negatives (FN)** refer to ground truth objects not detected by the model, while **True Negatives (TN)** are correctly classified background detections [14]. These classifications allow for the calculation of the following metrics:

- **Precision**: The ratio of True Positives to the total number of detections, measuring the model's ability to avoid false positives:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.1}$$

- **Recall**: The ratio of True Positives to the total number of ground-truth objects, measuring the model's ability to avoid false negatives:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.2}$$

Common metrics used to evaluate Object Detection include:

- **Average Precision (AP)**: This combines precision and recall, providing a single figure summarizing the model's performance across different confidence thresholds. Common versions are AP@.5 (with a threshold of 0.5 IoU) and AP@[.5:.05:.95], which calculates the average of AP values over several IoU thresholds [14].
- **Average Recall (AR)**: This measures the recall of the model averaged across multiple IoU thresholds. It can be computed for different numbers of detections per image, such as AR@1, AR@10, etc. [14].
- **Inference Time**: The time taken by the model to process an image, which is critical for applications requiring real-time detection [14].
- **Model Size**: The number of parameters or the size of the model, affecting deployment, especially on devices with limited resources [14].

- **Efficiency**: This considers the trade-off between accuracy and speed, often visualized using the AP vs. inference time curve [14].

In addition to Object Detection, Object Tracking is another critical task in Computer Vision, involving the continuous monitoring of objects across video frames. Object Tracking methods can be broadly classified into two categories: **Generative Trackers** and **Discriminative Trackers** [9]. Generative trackers are capable of handling challenging scenarios such as occlusion and large-scale variation through particle sampling strategies, often integrated with various appearance models, including sparse representation and energy of motion. Discriminative trackers, by contrast, build robust classifiers using hand-crafted or deep features [9]. The combination of generative and discriminative approaches, as well as the integration of deep learning techniques such as fully revolutionary networks and Transformer models, has led to significant improvements in object detection performance [32, 53, 52]. In addition, the speed and computational requirements of these algorithms are critical factors influencing their practical applicability [53, 52, 27]. Advanced techniques in object tracking leverage both generative and discriminative models to amplify tracking efficacy. The utilisation of deep trackers has evidenced superior results on public tracking datasets, attributed to their potent feature extractors, accurate bounding box regressors, and discriminative classifiers [27]. Techniques such as deformable convolution and Transformer models extend traditional convolution or correlation methodologies to execute global feature matching, thereby enhancing tracking accuracy. The incorporation of contextual or knowledge information can substantially elevate performance, with methodologies like Particle Filtering, also recognised as Sequential Monte Carlo (SMC) methods, framed as problems of Bayesian inference in state space [11, 46]. The extended Kalman Filtering (EKF) is another advanced technique that has been employed to improve tracking accuracy by predicting the current status through the previous status and modifying the prediction result based on observation information [53, 52]. Despite these advancements, the integration of these methods in a complementary manner remains an open research area with substantial potential for advancing the field [32, 53].

2.3 Deep Learning for Spacio-Temporal Prediction

Time series prediction involves processing sequential data to predict future events or values. Various deep learning models have been applied to this task, requiring several preparatory steps such as collecting data, designating attribute types, dealing with inconsistencies and storing datasets. These datasets are usually classified into units of time such as seconds, minutes and hours, allowing the construction of metadata for machine learning [28].

Deep Learning Sequence Models

The problem of predicting the future locations of objects has been extensively studied, particularly for static surveillance cameras. Initial efforts utilised recurrent neural networks (RNNs), including long-term memory networks (LSTMs) and gated recurrent units (GRUs), in an encoder-decoder format to encode past observations and decode future locations.

Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory networks (LSTMs) and Gated Recurrent Units (GRUs), have exhibited promise in this direction. Nevertheless, most RNN-based models suffer from performance degradation over time since they rely on recurrently predicting future bounding boxes based on previous outputs.

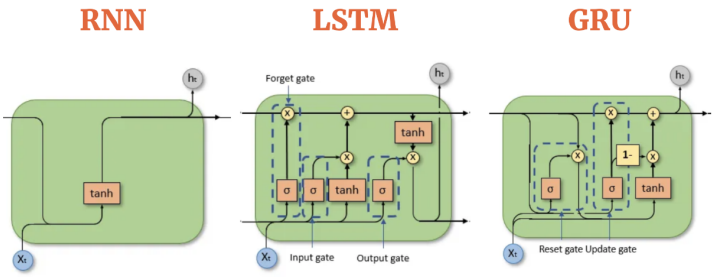


Figure 2.14: Comparing different Sequence models: RNN, LSTM, and GRU. Source: Colah's blog. Compiled by AIML.com

Early models integrated additional inputs such as environmental data and semantic actions to enhance prediction accuracy. For instance, Alahi et al. [1] proposed a Social-LSTM to model pedestrian trajectories and interactions, further improving global context capture through a social pooling module.

Related Work

STED Model

The STED (Spatio-Temporal Encoder-Decoder) model was introduced by Styles et al. [41] as a novel approach for multiple object forecasting (MOF), particularly in predicting the future bounding boxes of tracked objects from video sequences. This model is designed to handle the challenges of object forecasting in diverse environments, leveraging both visual and temporal features to predict object-motion and ego-motion effectively. As shown in figure 2.15, the STED model consists of three main components: a bounding box feature encoder, an optical flow feature encoder, and a decoder. The *Bounding Box Feature Encoder* utilises a Gated Recurrent Unit (GRU) to extract temporal features from past object bounding boxes, which

include coordinates (x, y), dimensions (w, h), and velocity changes $(\Delta x, \Delta y, \Delta w, \Delta h)$ over a window of 30 frames, representing 1 second of observation. Simultaneously, the *Optical Flow Feature Encoder* captures motion features directly from optical flow, using a Convolutional Neural Network (CNN) to process a stack of 10 frames sampled uniformly from the past 1 second of video data. The combination of these features provides a comprehensive understanding of both the object's movement and the camera's movement (ego-motion). The *Decoder* then takes the concatenated feature vector from the encoders and predicts the future bounding box coordinates for the next 60 frames (2 seconds prediction window). The GRU-based decoder generates bounding box predictions iteratively, using the encoded feature vector and the internal hidden state to output changes in bounding box velocity and dimensions over time.

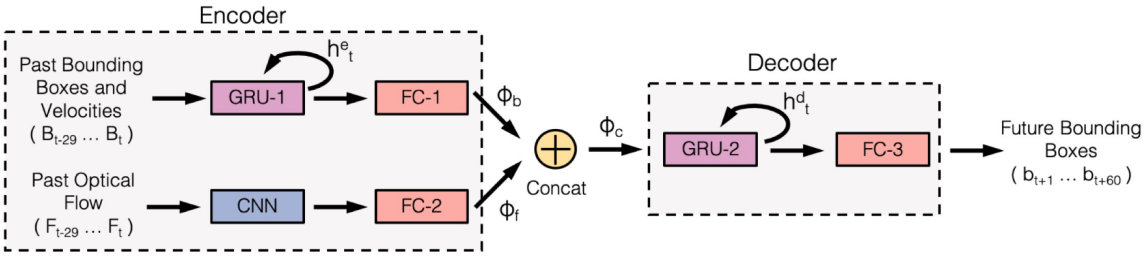


Figure 2.15: STED Model Architecture. Source: Styles et al. [41]

The STED architecture is specifically designed to address the complexities of predicting future object positions in video sequences, particularly under conditions of non-linear object motion and varying scales due to camera movement. By combining temporal features from bounding boxes with motion information from optical flow, the model effectively captures both the dynamic behavior of the objects and the impact of the camera's motion on the observed scene. The STED model was evaluated on the Citywalks dataset, a diverse dataset designed to test the model's ability to predict future object locations across different environments. The performance of STED was compared with several baseline models, as summarised in Table 2.2.

Table 2.2: Comparison of the performance of STED with baseline models on the Citywalks dataset. Metrics include Average Displacement Error (ADE), Final Displacement Error (FDE), Average Intersection-over-Union (AIOU), and Final Intersection-over-Union (FIOU). The model was evaluated using 1 second of input frames to predict 2 seconds of future frames.

Model	**ADE (pixels)**	**FDE (pixels)**	**AIOU (%)**	**FIOU (%)**
CV-CS [41]	31.6	57.6	46.0	21.3
LKF [41]	32.9	59.0	43.9	20.1
STED [41]	**26.0**	**46.9**	**51.8**	**27.5**

The STED model achieved an Average Displacement Error (ADE) of 26.0 pixels and a Final Displacement Error (FDE) of 46.9 pixels, with an Average Intersection Over Union (AIOU) of 51.8% and a Final Intersection Over Union (FIOU) of 27.5%. These results indicate that STED outperforms existing models in both displacement and intersection-over-union metrics, making it a robust solution for forecasting future object locations in challenging video sequences.

In summary, the STED model's ability to combine visual and temporal features from both object bounding boxes and optical flow enables it to predict future object positions more accurately, making it an effective tool for applications in autonomous driving and robotic navigation.

PV-LSTM Model

The PV-LSTM (Position-Velocity Long Short-Term Memory) model was developed by Bouhsain et al. [7] to predict pedestrian intentions and future bounding box positions, a crucial task for enhancing the safety of autonomous driving systems. The architecture of this model is illustrated in figure 2.16. The model utilises a sequence-to-sequence LSTM architecture that processes both spatial and temporal information effectively. To do that, it processes input sequences of observed bounding boxes (x, y, w, h) and their velocity changes $(\Delta x, \Delta y, \Delta w, \Delta h)$ over a window of 30 frames, representing 1 second of observation. These inputs are encoded through two encoders: the *Bounding Box Position Encoder* and the *Bounding Box Velocity Encoder*. The encoded features are then concatenated and passed to two decoders: the *Velocity Decoder*, which predicts future bounding box velocities, and the *Intention Decoder*, which predicts pedestrian crossing intentions over the next 60 frames (2 seconds prediction window). The PV-LSTM model's architecture is designed to capture the dynamic and complex nature of pedestrian behavior, which is crucial for real-time decision-making in autonomous driving. By integrating both position and velocity information through dual encoders, the model can more accurately predict the future movements and intentions of pedestrians, thus enhancing safety and reliability in autonomous navigation systems.

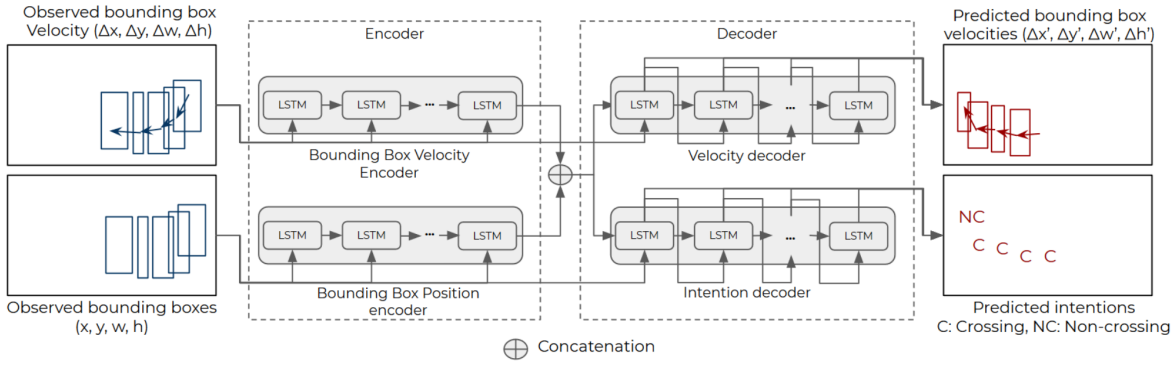


Figure 2.16: PV-LSTM Model Architecture. Source: Bouhsain et al. [7]

The PV-LSTM model was evaluated on the Citywalks dataset, and its performance was compared against several baseline models. The results are summarised in Table 2.3, showcasing the model's effectiveness in predicting pedestrian trajectories and intentions. The PV-LSTM model achieved an Average Displacement Error (ADE) of 25.2 pixels and a Final Displacement Error (FDE) of 49.9 pixels, with an Average Intersection Over Union (AIOU) of 40.2%. Although the STED model slightly outperforms in AIOU and FIOU, the PV-LSTM model provides a robust solution with competitive prediction accuracy, while maintaining a simpler architecture.

Table 2.3: Comparison of the performance of PV-LSTM with baseline models on the Citywalks dataset. Metrics include Average Displacement Error (ADE), Final Displacement Error (FDE), and Average Intersection-over-Union (AIOU). The model was evaluated using 1 second of input frames to predict 2 seconds of future frames.

Model	ADE (pixels)	FDE (pixels)	AIOU (%)	FIOU (%)
CV-CS [7]	31.6	57.6	46.0	21.3
LKF [7]	32.9	59.0	43.9	20.1
STED [7]	**26.0**	**46.9**	**51.8**	**27.5**
PV-LSTM [7]	25.2	49.9	40.2	20.3

Fusion-GRU Model

Karim et al. [25] developed the Fusion-Gated Recurrent Unit (Fusion-GRU) model to predict the future bounding boxes of traffic agents in risky driving scenarios. This model leverages multiple sources of information, such as location-scale data, monocular depth information, and optical flow data, to capture complex interactions among these cues and transform them into meaningful hidden representations. This approach is particularly suited for dealing with scenarios where the available observation time is limited due to abrupt motion changes or tracking loss. The Fusion-GRU model consists of several components that work together to predict future bounding boxes. The model takes input video sequences, extracts depth maps, and calculates optical flow to capture motion dynamics. Bounding boxes are detected and tracked using YOLOv5 and DeepSort, respectively. The feature extractor processes both object-level and scene-level features using ResNet50, transforming them into feature vectors. These vectors are then concatenated and passed to the Fusion-GRU encoder, which integrates location, scale, and distance information into hidden representations. The model also introduces an intermediary estimator, which generates intermediate bounding boxes that help in capturing sequential dependencies across frames. Additionally, a self-attention aggregation layer is employed to reduce error accumulation by focusing on the most relevant information for long-term predictions. Finally, a GRU decoder iteratively predicts the future bounding boxes based on the aggregated feature vectors and the hidden representations from the Fusion-GRU encoder.

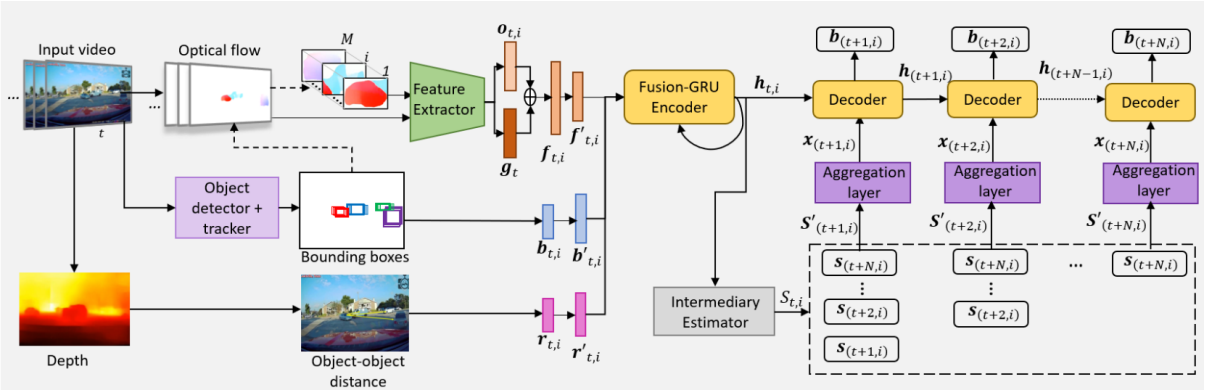


Figure 2.17: Fusion-GRU model architecture. Source: Karim et al. [25]

The Fusion-GRU model is specifically designed to address the limitations of traditional GRU-based models, which often suffer from performance degradation over time. By integrating complex interactions among various input features and employing self-attention mecha-

nisms, the Fusion-GRU model enhances its ability to predict future bounding boxes more accurately. This architecture is particularly effective in cluttered and dynamic driving environments, where understanding the relationships between different traffic agents and their surroundings is crucial for accurate prediction.

The Fusion-GRU model was evaluated on two publicly available datasets, ROL (Risky Object Localization) and HEV-I (Honda Egocentric View-Intersection), demonstrating its superior performance in predicting future bounding boxes compared to other state-of-the-art models. The results are summarised in Table 5.2, which compares the performance of Fusion-GRU with baseline models based on metrics such as Average Displacement Error (ADE), Final Displacement Error (FDE), and Intersection over Union (IOU).

Table 2.4: Comparison of the performance of Fusion-GRU with baseline models on the ROL and HEV-I datasets. Metrics include ADE, FDE, and FIoU. The model was evaluated using 0.5-second and 1-second prediction horizons.

Model	**ADE0.5 (pixels)**	**FDE0.5 (pixels)**	**FIoU0.5 (%)**	**ADE1.0 (pixels)**
FOL-X [25]	6.7	11.0	85.0	12.6
SGDNet [25]	6.3	–	–	11.4
Fusion-GRU [25]	**5.5**	**8.3**	**85.2**	**11.2**

The Fusion-GRU model achieved the lowest Average Displacement Error (ADE) and Final Displacement Error (FDE) in both 0.5-second and 1-second prediction horizons on the HEV-I dataset. It also exhibited the highest Final Intersection over Union (FIoU) of 85.2% in the 0.5-second horizon, indicating its strong capability to accurately predict the future positions and scales of traffic agents.

Chapter 3

Methodology

3.1 Dataset Configuration

3.1.1 Provided Dataset Description

The 'Indoor Hangar Fueling Port Detection and Tracking Dataset' (HARD) is an integral part of Phase-1 of the ONEHeart project, funded by UKRI. The ONEHeart project aims to develop an automated aircraft refuelling system based on computer vision and robotics technology. This dataset can be utilised for various purposes, including but not limited to refueling port detection, fueling port tracking, camera pose estimation, and visual image processing. It is provided under the terms of the UKRI funding agreement. Unauthorised use, distribution, or reproduction is prohibited. The HARD dataset consists of 21 video sequences captured in an indoor hangar at the Aerospace Integration Research Centre (AIRC). The videos were recorded using an Intel® RealSense™ D435 [21] (Shown in Figure 3.1), which provides depth information in addition to RGB data. The target area for detection and tracking is near the refueling port of an Airbus A320 wing. The videos were recorded under different lighting conditions, including indoor artificial lighting and natural daylight, to simulate real-world scenarios. In addition, the refueling port was in different states (closed, open, and semi-open) to capture a wide range of variations for training and testing machine learning models.

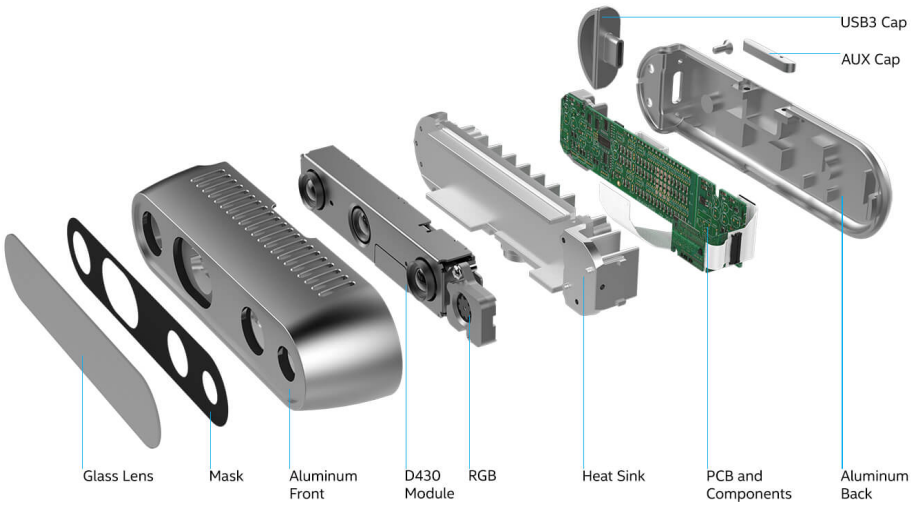


Figure 3.1: Intel® RealSense™ D435 Depth Camera. Source: Intel

3.1.2 Data Annotation

The HARD dataset provided for this project was not fully annotated. Therefore, the first step in the data preparation process was to annotate the dataset. This annotation process was crucial to enable the training of machine learning models for accurate refueling port detection and tracking. Initially, 100 frames from each video sequence were manually annotated. This involved labeling the refueling port in each frame, which required identifying and marking the exact location of the refueling port using bounding boxes. This was done using the Label Studio tool, a powerful annotation platform that allows users to create and manage annotations in images and videos. Label Studio was chosen for its flexibility and ease of use. It supports various annotation formats and integrates well with machine learning workflows. Users can draw bounding boxes around objects of interest, in this case, the refueling port, to create labeled datasets. The initial manual annotations created a preliminary dataset. This dataset was used to train a YOLOv10 (You Only Look Once, version 10) model for refueling port detection. The annotated dataset was used to train the YOLOv10 model. The model learned to detect the refueling port from the annotated images, improving its accuracy with each training iteration. After training the YOLOv10 model, it was implemented as a backend service for Label Studio. This was deployed as a Docker container, allowing the model to be used for automated annotation of the remaining images in the dataset. The model predicted the location of the refueling port in new frames, and these predictions were used to annotate the rest of the dataset automatically. Each automatically generated annotation was reviewed manually to ensure accuracy. This review process involved verifying the location of the refueling port in each frame and making necessary corrections to the annotations. This thorough quality control ensured that the entire dataset was consistently and accurately labeled.This comprehensive annotation process ensured that the entire dataset was accurately labeled, providing a robust foundation for training machine learning models aimed at refueling port detection and tracking. The combination of manual and automated annotation techniques maximised efficiency while maintaining high annotation quality.

3.1.3 Summary of Available Videos

Each video was assigned to a specific dataset (train, val, and test), with an approximate split of 70% for training, 15% for validation, and 15% for testing. The following table presents the available videos in the dataset along with the number of frames for each video and their assignment:

Type	Video Name	Number of Frames	Assignment
Closed	**video_lab_platform_1**	**624**	**train**
Closed	video_lab_platform_2	639	train
Closed	video_lab_platform_5	398	train
Closed	video_lab_platform_7	412	train
Closed	video_lab_platform_8	470	train
Closed	video_lab_platform_9	373	train
Closed	video_lab_manual_1	746	train
Closed	video_lab_platform_3	569	val
Closed	video_lab_platform_4	247	val
Closed	video_lab_platform_6	303	test
Closed	test_outdoor1	499	test
Open	**video_lab_open_1_____1**	**497**	**train**
Open	video_lab_open_1_____2	602	train
Open	video_lab_open_1_____3	313	train
Open	video_lab_open_1_____4	310	train
Open	test_indoor2	310	val
Open	test_indoor1	314	test
Semi-Open	**video_lab_semiopen_1_____1**	**739**	**train**
Semi-Open	video_lab_semiopen_1_____2	439	train
Semi-Open	video_lab_semiopen_1_____4	372	val
Semi-Open	video_lab_semiopen_1_____3	383	test

Table 3.1: Summary of available videos in the HARD dataset with their assignment.

3.1.4 Initial Data Distribution

Before balancing the data, the distribution of video frames across the different datasets (train, validation, and test) for each state of the fueling port (CLOSED, OPEN, and SEMI-OPEN) was as follows:

Type	Total Frames	Train	Test	Validation
CLOSED	5280	3662 (69.36%)	802 (15.19%)	816 (15.45%)
OPEN	2346	1722 (73.40%)	314 (13.38%)	310 (13.21%)
SEMI-OPEN	1933	1178 (60.94%)	383 (19.81%)	372 (19.24%)

Table 3.2: Distribution of frames across train, test, and validation sets for each state in the HARD dataset before balancing.

As shown in 3.2, the dataset initially had an imbalance in the number of frames for each state. For instance, the CLOSED state had significantly more frames compared to the OPEN

and SEMI-OPEN states. This imbalance could lead to biased training and inaccurate model performance, as the model might become overly familiar with the more prevalent CLOSED state and underperform on the less represented states.

3.1.5 Balanced Data Distribution

To address this issue, a balancing strategy was employed to create a more uniform dataset. The first step involved shuffling and subsetting each state (CLOSED, OPEN, and SEMI-OPEN) to keep only the minimum number of frames for each state. This ensured that the number of frames for each state was equal, avoiding bias towards any particular state. The subsets were then merged to create three balanced datasets: Training, Validation, and Test, each with an equal number of frames for each state. Finally, the balanced datasets were shuffled again and resized to maintain the required split ratio of 70% for training, 15% for validation, and 15% for testing. The balanced dataset will be used for training and evaluating the object detection model, while the full dataset will be utilised for sequence model training and framework evaluation. The resulting distribution of frames across the train, test, and validation sets for each state after balancing is as follows:

Dataset	Total Frames
Train	3534 (69.57%)
Test	773 (15.22%)
Val	773 (15.22%)

Table 3.3: Distribution of frames across train, test, and validation sets for each state in the HARD dataset after balancing.

The primary reason for balancing the dataset is to ensure that the object detection model is equally trained on all states of the refueling port. An imbalanced dataset could cause the model to perform well on the more common states (like CLOSED) while underperforming on the less common states (like OPEN and SEMI-OPEN). By balancing the dataset, the model has an equal opportunity to learn from all states, improving its generalisation and robustness. This strategy also prevents the model from becoming biased towards any particular state, ensuring consistent performance across all states, which is critical for the reliable operation of the automated refueling system.

3.1.6 Example Images from the Dataset

The following figures show annotated examples of the refueling port in different states:

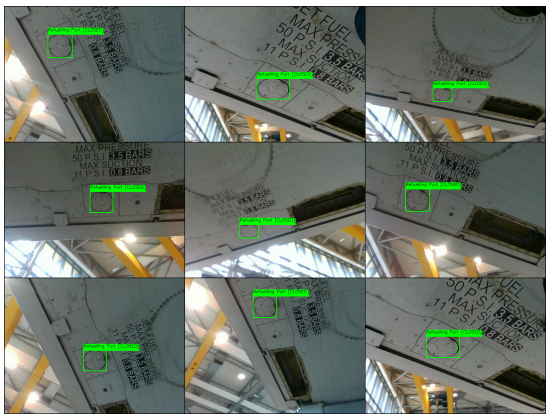


Figure 3.2: Annotated images of the refueling port in the CLOSED state.

Figure 3.3: Annotated images of the refueling port in the OPEN state.

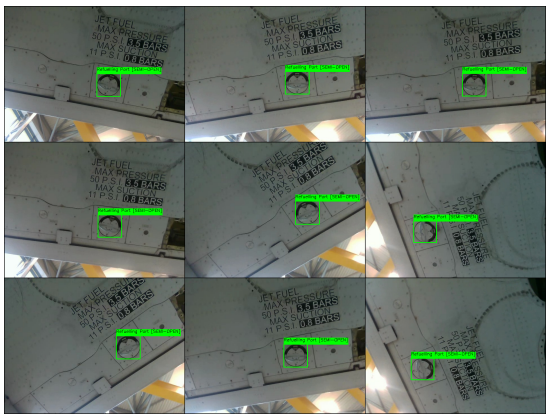


Figure 3.4: Annotated images of the refueling port in the SEMI-OPEN state.

3.2 Framework Design

The proposed framework, illustrated in Figure 3.5, is designed to predict the future positions of the refueling port's bounding boxes across the next m frames of a video stream. It does this by leveraging the bounding boxes observed in the current and previous frames. At any given time step t, the observed bounding box is denoted as $\mathbf{b}_t = [x_t, y_t, w_t, h_t]$, where (x_t, y_t) represents the center coordinates of the bounding box, and w_t and h_t represent the width and height of the bounding box, respectively. The goal is to predict a sequence of future bounding boxes, $\mathbf{S}_{t+1:t+m} = \{\mathbf{b}_{t+1}, \ldots, \mathbf{b}_{t+m}\}$, for the upcoming m frames based on the observed bounding boxes from the past n frames, $\mathbf{S}_{t-1+n:t} = \{\mathbf{b}_{t-1+n}, \ldots, \mathbf{b}_t\}$.

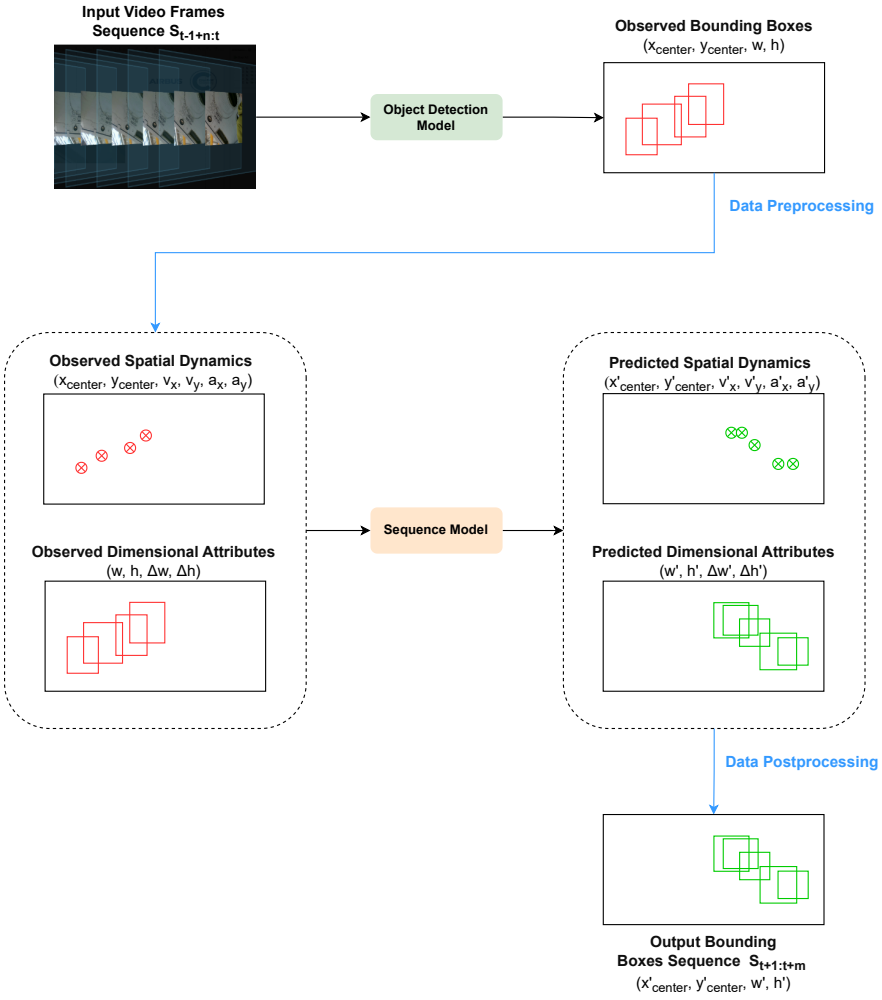


Figure 3.5: Framework Workflow

The process starts with an input video sequence, which is analysed on a frame-by-frame basis. An object detection model first identifies and locates the bounding boxes of the refueling port in each frame. These detected bounding boxes are then formatted into input vectors suitable for the sequence model. The sequence model is responsible for predicting the future spatial positions and dimensions of the refueling port's bounding boxes. Finally, these predictions are refined through postprocessing steps to produce the final bounding box forecasts for the subsequent frames.

3.3 Object Detection Model Fine-tuning

A fine-tuned YOLOv10 Nano model is used to detect the refueling port within the video frames. As highlighted in the literature review, YOLOv10 represents the latest advancement in object detection technology, offering an optimal balance between speed and accuracy. The YOLOv10 Nano variant is particularly well-suited for real-time applications, especially on resource-constrained embedded devices, due to its lightweight architecture. The model is fine-tuned through transfer learning on the HARD dataset, enabling precise detection of the refueling port within the video frames. This fine-tuning process ensures that the model meets the demands of real-time video analysis with both accuracy and responsiveness. The implementation uses the Ultralytics YOLOv10 framework, which provides a reliable and efficient pipeline for both training and inference. This framework optimises detection performance and guarantees robust operation in different scenarios.

3.4 Sequence Model Design

This thesis introduces the *SizPos-GRU model* model, a deep learning sequence model for predicting future positions and sizes of a unique object in a video stream. The model leverages an encoder-attention-decoder architecture to capture temporal dependencies and spatial relationships effectively. The SizPos-GRU model leverages an encoder-attention-decoder architecture to effectively capture temporal dependencies and spatial relationships. The framework consists of four main components: two encoders, an attention mechanism, and two decoders. Each component is designed to handle specific aspects of the sequence modeling task, from encoding input sequences to generating context-aware predictions.

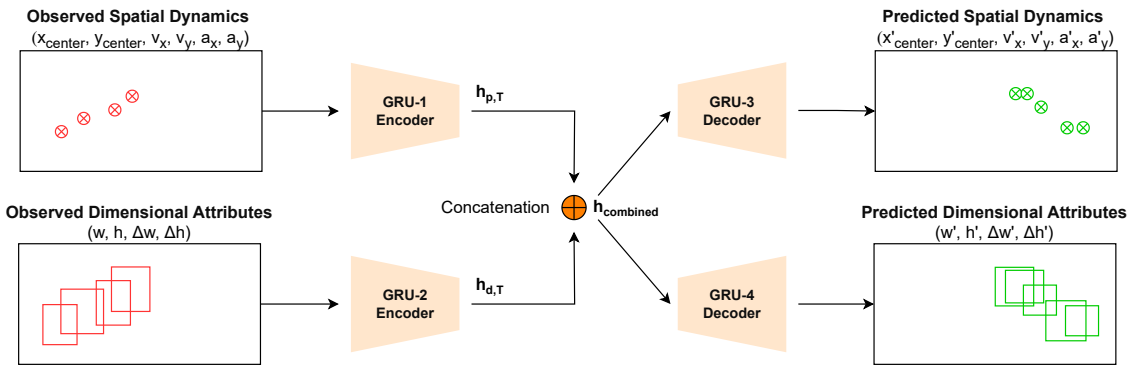


Figure 3.6: SizPos-GRU Model Architecture

3.4.1 Input Representation

The *SizPos-GRU* model expects two types of input sequences to predict the future positions and sizes of an object. The first input captures the spatial dynamics of the object, including its position, velocity and acceleration. The second input focuses on the object's dimensional attributes, such as its width and height, and changes in these dimensions over time.

Spatial Dynamics Vector

The spatial dynamics vector at time t, denoted as $\mathbf{p}_t$, represents the position, velocity, and acceleration of the bounding box center. It is defined as:

$$\mathbf{p}_t = (x_t, y_t, v_{x,t}, v_{y,t}, a_{x,t}, a_{y,t}), \tag{3.1}$$

$$v_{x,t} = x_t - x_{t-1}, \quad v_{y,t} = y_t - y_{t-1}, \tag{3.2}$$

$$a_{x,t} = v_{x,t} - v_{x,t-1}, \quad a_{y,t} = v_{y,t} - v_{y,t-1}. \tag{3.3}$$

Here, (x_t, y_t) represents the center coordinates at time t, $(v_{x,t}, v_{y,t})$ are the velocities, and $(a_{x,t}, a_{y,t})$ are the accelerations along the x and y axes, respectively.

Dimensional Attributes Vector

The dimensional attributes vector at time t, denoted as $\mathbf{d}_t$, captures the size of the bounding box and the changes in its dimensions:

$$\mathbf{d}_t = (w_t, h_t, \Delta w_t, \Delta h_t), \tag{3.4}$$

$$\Delta w_t = w_t - w_{t-1}, \quad \Delta h_t = h_t - h_{t-1}. \tag{3.5}$$

Here, (w_t, h_t) represents the width and height at time t, and $(\Delta w_t, \Delta h_t)$ are the changes in these dimensions from the previous time step.

Input Sequences for Model

The sequences of these vectors over a time window of n frames are fed into the model as:

$$\mathbf{P} = \{\mathbf{p}_{t-1+n}, \mathbf{p}_{t-n+2}, \ldots, \mathbf{p}_t\}, \quad \mathbf{D} = \{\mathbf{d}_{t-1+n}, \mathbf{d}_{t-n+2}, \ldots, \mathbf{d}_t\},$$

where $\mathbf{P}$ is the sequence of spatial dynamics vectors, and $\mathbf{D}$ is the sequence of dimensional attributes vectors. These sequences are processed by the model's encoders to extract temporal features, which are then used to predict future bounding box positions and sizes.

3.4.2 Encoders

The SizPos-GRU model employs two separate GRU encoders: one for processing the sequence of spatial dynamics vectors and another for processing the sequence of dimensional attributes vectors. Both encoders are designed to extract meaningful temporal features from their respective input sequences, which are subsequently used by the decoders to generate predictions.

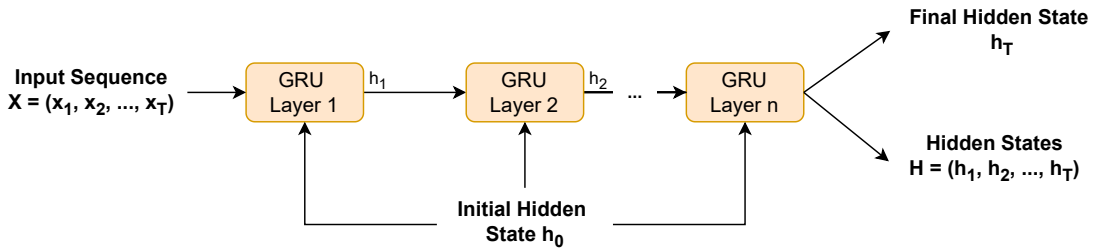


Figure 3.7: SizPos-GRU Encoder Architecture. The input sequence $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\}$ represents either the spatial dynamics vector ($\mathbf{P}$) or the dimensional attributes vector ($\mathbf{D}$). This sequence is processed through multiple GRU layers, producing a sequence of hidden states $H = \{h_1, h_2, \ldots, h_T\}$ and a final hidden state h_T that encapsulates the temporal dependencies in the input sequence.

Position Encoder

The position encoder processes the sequence of spatial dynamics vectors, which include the bounding box's position, velocity, and acceleration over time. It is defined as:

$$H_p, h_{p_T} = \text{GRU}_p(\mathbf{P}, h_{p_0}), \tag{3.6}$$

where $\mathbf{P} = (\mathbf{p}_{t-1+n}, \mathbf{p}_{t-n+2}, \ldots, \mathbf{p}_t)$ is the input sequence of spatial dynamics vectors, $H_p = (h_{p_1}, h_{p_2}, \ldots, h_{p_T})$ represents the sequence of hidden states generated by the GRU, and h_{p_T} is the final hidden state at time T. The initial hidden state h_{p_0} is typically initialised to zero.

Size Encoder

The size encoder processes the sequence of dimensional attributes vectors, which include the bounding box's width, height, and changes in these dimensions over time. It is defined as:

$$H_d, h_{d_T} = \text{GRU}_d(\mathbf{D}, h_{d_0}), \tag{3.7}$$

where $\mathbf{D} = (\mathbf{d}_{t-1+n}, \mathbf{d}_{t-n+2}, \ldots, \mathbf{d}_t)$ is the input sequence of dimensional attributes vectors, $H_d = (h_{d_1}, h_{d_2}, \ldots, h_{d_T})$ represents the sequence of hidden states generated by the GRU, and h_{d_T} is the final hidden state at time T. Similarly, the initial hidden state h_{d_0} is initialised to zero.

3.4.3 Hidden State Fusion

After the position and size encoders process their respective input sequences, the resulting hidden states are combined to leverage information from both spatial dynamics and dimensional attributes. This fusion is achieved by concatenating the final hidden states from both encoders and passing them through a fully connected layer:

$$h_{\text{combined}} = W_{\text{combine}} \cdot [h_{p_T} \oplus h_{d_T}] + b_{\text{combine}}, \tag{3.8}$$

where h_{p_T} and h_{d_T} are the final hidden states from the position and size encoders, respectively, $\oplus$ denotes the concatenation operation, W_{combine} is the weight matrix, and b_{combine} is the bias vector. The fused hidden state h_{combined} encapsulates the temporal features from both input sequences, enabling the model to generate context-aware predictions.

3.4.4 Decoders

The *SizPos-GRU* model utilises two specialised decoders to predict the future bounding boxes over the next m frames. These decoders are designed to process the temporal features extracted by the encoders and the fused hidden states, generating accurate and context-aware predictions.

3.4.4.1 Position Decoder

The position decoder is responsible for predicting the future spatial dynamics of the bounding boxes, including their positions and velocities. The decoding process begins by initialising the input with the last observed position in the sequence. The decoder then iteratively predicts future positions over the defined prediction horizon m.

During each iteration, the current input is processed through a stack of GRU layers, which update the hidden state based on the previous hidden state and current input. This updated hidden state, encapsulating temporal dependencies, is then passed through a self-attention mechanism. The self-attention mechanism computes attention scores, which are normalised using a softmax function to produce attention weights. These weights are applied to the hidden states to generate a context vector that highlights the most relevant temporal features.

$$H_p = \text{GRU}p(h\text{combined}), \tag{3.9}$$

$$\alpha_i = \frac{\exp(w_i)}{\sum_{j=1}^{m} \exp(w_j)}, \quad w_i = \text{Linear}(h_{p_i}), \tag{3.10}$$

$$c_p = \sum_{i=1}^{m} \alpha_i h_{p_i}, \tag{3.11}$$

The context vector c_p is then passed through a series of fully connected layers, with dropout and ReLU activation functions, to produce the final position prediction for the current time step:

$$\hat{\mathbf{P}}_{t+1:m} = \text{ReLU}(\text{Dropout}(\text{Dense1}(c_p))), \tag{3.12}$$

$$\hat{\mathbf{P}}t+1:m = \text{Dense2}(\hat{\mathbf{P}}t+1:m), \tag{3.13}$$

The predicted position is then used as the input for the next iteration of the decoding loop, enabling the model to recursively generate predictions for subsequent time steps. This iterative process continues until predictions for all future time steps within the prediction window have been generated, and the accumulated sequence of predicted positions is returned as the final output.

3.4.4.2 Size Decoder

The size decoder operates similarly to the position decoder but focuses on predicting the future dimensions of the bounding boxes, such as width and height. The decoding process starts with the last observed size in the sequence, and the decoder iteratively generates predictions for future sizes over the prediction horizon m.

In each iteration, the current size input is processed through GRU layers that capture the temporal dynamics of size changes. The output from the GRU is then passed through a self-attention mechanism, which assigns different weights to the hidden states across time. These weights are used to compute a context vector c_d, which represents the most relevant temporal and spatial information for the current prediction step.

$$H_d = \text{GRU}d(h\text{combined}), \tag{3.14}$$

$$\beta_i = \frac{\exp(v_i)}{\sum_{j=1}^{m} \exp(v_j)}, \quad v_i = \text{Linear}(h_{d_i}), \tag{3.15}$$

$$c_d = \sum_{i=1}^{m} \beta_i h_{d_i}, \tag{3.16}$$

The context vector c_d is passed through dense layers with dropout and ReLU activation to produce the final size prediction:

$$\hat{\mathbf{D}}_{t+1:m} = \text{ReLU}(\text{Dropout}(\text{Dense1}(c_d))), \tag{3.17}$$

$$\hat{\mathbf{D}}t + 1 : m = \text{Dense2}(\hat{\mathbf{D}}t + 1 : m), \tag{3.18}$$

Similar to the position decoder, the predicted size is then used as input for the next iteration, ensuring that each prediction is contextually informed by previous outputs. This loop continues until the decoder has generated predictions for all future time steps within the prediction window m, which are then returned as the sequence of predicted sizes.

3.4.4.3 Self-Attention Mechanism

The self-attention mechanism plays a pivotal role in both decoders by enhancing the model's ability to focus on the most relevant parts of the input sequences. By dynamically assigning weights to different time steps, the self-attention mechanism allows the model to capture long-range dependencies, which are crucial for accurate predictions over extended time horizons.

3.4.4.4 Final Output

The final output of the decoding process consists of sequences of predicted positions and sizes for the specified number of future time steps m. These predictions provide valuable insights into the expected future states of the bounding boxes, including their movement and changes in dimensions. The integration of GRU layers and self-attention mechanisms within the decoders ensures that the *SizPos-GRU* model can effectively handle complex sequence prediction tasks, making it well-suited for applications such as video analysis and object tracking.

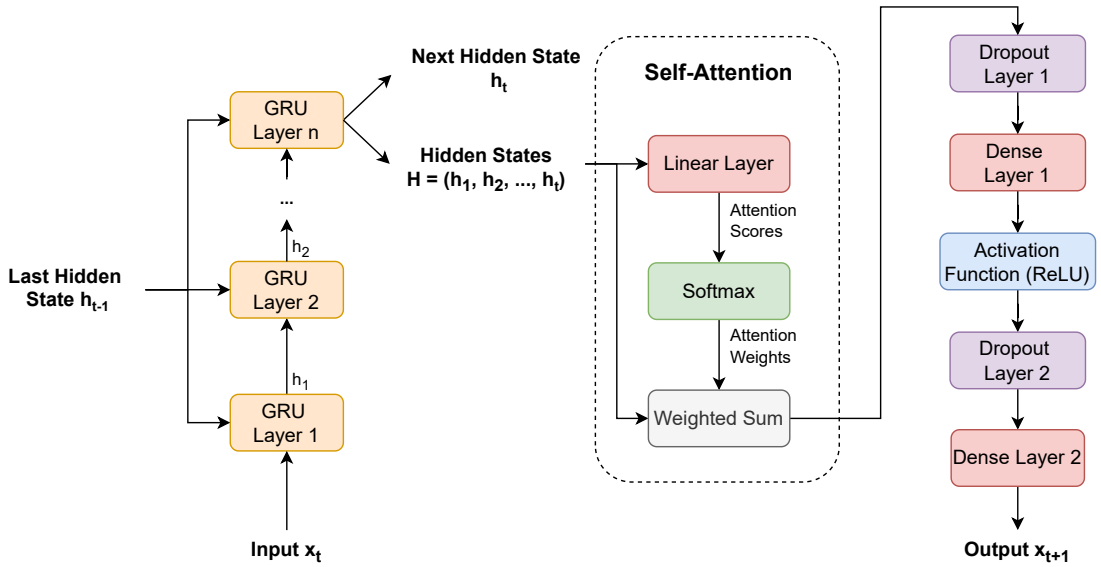


Figure 3.8: SizPos-GRU Decoder Architecture. This architecture illustrates the decoding process where the input sequence $\mathbf{x}_t$ and the last hidden state h_{t-1} are processed through multiple GRU layers to generate the next hidden state h_t. The sequence of hidden states $H = \{h_1, h_2, \ldots, h_t\}$ is then passed through a self-attention mechanism, which calculates attention scores and weights. The weighted sum of hidden states is combined with linear and non-linear transformations, including dropout and ReLU activation functions, to produce the final output $\mathbf{x}_{t+1}$. This output is used for predicting the next time step in the sequence, continuing the process iteratively for future predictions.

3.5 Algorithm Design

3.5.1 Data Preprocessing

3.5.2 Data Postprocessing

3.5.3 Data Augmentation Strategy

The data augmentation strategy is designed to enhance the robustness and generalization capability of the model. This strategy simulates various camera movements and imperfections that might occur in real-world scenarios. The augmentation approach consists of the following key components:

Sequence Reversal

For the training stage, the size of the dataset is doubled by including reversed sequences:

- For each original sequence $(S_1, S_2, ..., S_n)$, a reversed sequence $(S_n, S_{n-1}, ..., S_1)$ is added.
- This augmentation helps the model learn to predict both forward and backward movements of the camera relative to the aircraft refueling port.

Camera Movement Simulation

Subtle camera movements are simulated to make the model more robust to varying camera positions:

- **Panning**: A smooth, cumulative random offset is applied to the x and y coordinates of the bounding boxes. This simulates the effect of the camera panning horizontally or vertically.
- The panning effect is modeled as:

$$\text{pan}_t = \sum_{i=1}^{t} \frac{N(0, \sigma^2)}{5} \tag{3.19}$$

  where $N(0, \sigma^2)$ is a normal distribution with mean 0 and variance σ^2.

Zoom Simulation

To account for changes in the apparent size of the refueling port due to camera zoom or aircraft movement:

- A smooth, cumulative scaling factor is applied to the width and height of the bounding boxes.
- The zoom effect is modeled as:

$$\text{zoom}_t = \prod_{i=1}^{t} U(0.98, 1.02) \tag{3.20}$$

  where $U(0.98, 1.02)$ is a uniform distribution between 0.98 and 1.02.

Detection Inaccuracy Simulation

To make the model more robust to slight inaccuracies in object detection:

- Small Gaussian noise is added to all bounding box coordinates.
- The noise is sampled from $N(0, 0.002^2)$ for each coordinate.

3.5.4 Implementation Details

The augmentation strategy is applied with the following considerations:

- Augmentations are only applied during the training stage.
- Each sequence has a 50% chance of being augmented.
- When applied, augmentations affect both input and output sequences consistently to maintain temporal coherence.
- All augmented values are clipped to the range [0, 1] to ensure they remain valid normalised coordinates in the YOLO format.

This comprehensive augmentation strategy enhances the diversity of the training data, helping the model generalise better to various real-world scenarios involving camera movements and detection variations.

Chapter 4

Experiment Design

4.1 Experiment Environment

4.2 Comparison Experiments

To evaluate the performance of the proposed *SizPos-GRU* model, a series of comparison experiments are conducted using various baseline models. These models are designed to predict the future positions and sizes of the refueling port in video sequences, providing a reference point for assessing the effectiveness of the *SizPos-GRU* model.

Linear Kalman Filter (LKF)

The Linear Kalman Filter (LKF) is a powerful tool for predicting the future positions of objects based on their observed trajectories. This model is particularly useful in scenarios where the motion of objects follows a linear pattern with Gaussian noise. In this implementation, the LKF is applied to predict future bounding box positions in a video stream. The LKF operates on sequences of past bounding box positions. The state vector $\mathbf{x} \in \mathbb{R}^4$ includes the position and velocity components, and the state transition matrix $\mathbf{F}$ governs the linear motion model:

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

where Δt is the time step between frames.

Given a sequence of past positions $\mathbf{P} \in \mathbb{R}^{T \times 4}$, where T is the number of observed frames, the LKF iteratively predicts and updates the state estimate $\mathbf{x}$ using the following steps:

1. **Prediction Step:** The state and error covariance matrices are predicted using the state transition matrix $\mathbf{F}$ and process noise covariance $\mathbf{Q}$.

2. **Update Step:** The Kalman Gain $\mathbf{K}$ is computed, and the state estimate is updated using the latest measurement.

This process continues for the length of the observed sequence, and the final state is used to predict the future positions for up to K frames.

Constant Velocity (CV)

The Constant Velocity Trajectory Prediction Model assumes that an object's future trajectory can be extrapolated from its recent motion, with constant speed and direction. Given a sequence of past bounding boxes $\mathbf{P} \in \mathbb{R}^{T\times 4}$, where T is the number of past time steps, the model first computes the velocities between consecutive frames:

$$\text{velocities} = \mathbf{P}[:, 1:] - \mathbf{P}[:, :-1].$$

The average velocity $\bar{\mathbf{V}}$ is then calculated across all time steps:

$$\bar{\mathbf{V}} = \text{mean}(\text{velocities}, \text{dim} = 1, \text{keepdim=True}).$$

Using the last observed position $\mathbf{P}_T$ and the average velocity $\bar{\mathbf{V}}$, the future positions for $k = 1, 2, \ldots, K$ frames are predicted as:

$$\mathbf{F}_k = \mathbf{P}_T + k \cdot \bar{\mathbf{V}},$$

where K represents the number of future frames to predict. This prediction assumes that the object will continue to move with a constant velocity.

LSTM PosVelAcc

The LSTM PosVelAcc model employs a Long Short-Term Memory (LSTM) network to predict future bounding box positions by encoding past positions, velocities, and accelerations. The model is designed to capture the temporal dependencies in the movement of the refueling port. The input features include positional data, velocities, and accelerations, and the LSTM architecture is configured with multiple layers and hidden units. This model was inspired by the PV-LSTM model from Bouhsain et al. [7].

LSTM SizPos

The LSTM SizPos model is a variant of the LSTM architecture that focuses on predicting future bounding box sizes and positions. This model encodes both the size (width and height) and positional data of the bounding boxes to provide a comprehensive prediction. The LSTM SizPos model differs from the LSTM PosVelAcc model in its input representation and architecture, specifically designed to handle changes in both size and position over time. The predictions of size and position are combined to produce the final bounding box predictions for future frames.

GRU PosVelAcc

The GRU PosVelAcc model is similar to the LSTM PosVelAcc model but uses a Gated Recurrent Unit (GRU) network instead of an LSTM. GRU networks are known for their computational efficiency and are used here to predict future positions by leveraging past positional, velocity, and acceleration data. The GRU architecture is configured to capture the temporal dynamics of the bounding box, and its simplicity offers an efficient alternative to the LSTM-based models. This model was inspired by the Fusion-GRU model from Karim et al. [25].

4.3 Evaluation Metrics

To evaluate the performance of the proposed framework, four key metrics are used: Average Displacement Error (ADE), Final Displacement Error (FDE), Average Intersection over Union (AIOU), and Final Intersection over Union (FIOU). These metrics provide a comprehensive assessment of the accuracy and robustness of the predicted bounding boxes over time.

Average Displacement Error (ADE)

The Average Displacement Error (ADE) measures the average Euclidean distance between the predicted bounding box centers and the ground truth bounding box centers over all predicted frames. It is defined as:

$$\text{ADE} = \frac{1}{m}\sum_{t=1}^{m}\sqrt{(x_t^{\text{pred}} - x_t^{\text{gt}})^2 + (y_t^{\text{pred}} - y_t^{\text{gt}})^2}, \tag{4.1}$$

where m is the number of predicted frames, $(x_t^{\text{pred}}, y_t^{\text{pred}})$ represents the center of the predicted bounding box at the t-th frame, and $(x_t^{\text{gt}}, y_t^{\text{gt}})$ represents the center of the ground truth bounding box at the same frame.

Final Displacement Error (FDE)

The Final Displacement Error (FDE) focuses on the accuracy of the predicted bounding box center in the final frame of the prediction horizon. It is defined as the Euclidean distance between the predicted and ground truth bounding box centers at the final frame m:

$$\text{FDE} = \sqrt{(x_T^{\text{pred}} - x_T^{\text{gt}})^2 + (y_T^{\text{pred}} - y_T^{\text{gt}})^2}. \tag{4.2}$$

Average Intersection over Union (AIOU)

The Average Intersection over Union (AIOU) evaluates the overlap between the predicted bounding boxes and the ground truth bounding boxes, averaged over all predicted frames. It is computed as:

$$\text{AIOU} = \frac{1}{m}\sum_{t=1}^{m}\frac{A(\mathbf{b}_t^{\text{pred}} \cap \mathbf{b}_t^{\text{gt}})}{A(\mathbf{b}_t^{\text{pred}} \cup \mathbf{b}_t^{\text{gt}})}, \tag{4.3}$$

where $A(\mathbf{b}_t^{\text{pred}} \cap \mathbf{b}_t^{\text{gt}})$ denotes the area of the intersection between the predicted bounding box $\mathbf{b}_t^{\text{pred}}$ and the ground truth bounding box $\mathbf{b}_t^{\text{gt}}$ at the t-th frame, and $A(\mathbf{b}_t^{\text{pred}} \cup \mathbf{b}_t^{\text{gt}})$ denotes the area of their union.

Final Intersection over Union (FIOU)

The Final Intersection over Union (FIOU) assesses the overlap between the predicted bounding box and the ground truth bounding box at the final frame of the prediction horizon. It is defined as:

$$\text{FIOU} = \frac{A(\mathbf{b}_T^{\text{pred}} \cap \mathbf{b}_T^{\text{gt}})}{A(\mathbf{b}_T^{\text{pred}} \cup \mathbf{b}_T^{\text{gt}})}, \tag{4.4}$$

where $A(\mathbf{b}_T^{\text{pred}} \cap \mathbf{b}_T^{\text{gt}})$ and $A(\mathbf{b}_T^{\text{pred}} \cup \mathbf{b}_T^{\text{gt}})$ denote the intersection and union areas at the final frame m, respectively.

These metrics provide a balanced evaluation of the prediction performance, capturing both the spatial accuracy and the temporal consistency of the predicted bounding boxes.

Chapter 5

Results and Discussion

5.1 Object Detection Training Results

This section presents the final testing results of three different YOLO models (yolov10n.pt, yolov10s.pt, yolov10m.pt) on a dataset involving the classification of fuel port states (CLOSED, SEMI-OPEN, OPEN).

Table 5.1: Comparison of YOLO Models

Model	Metric	All	Fuel Port [CLOSED]	Fuel Port [SEMI-OPEN]	Fuel Port [OPEN]
YOLOv10n.pt	Precision (P)	0.989	0.994	0.982	0.992
	Recall (R)	0.876	0.657	0.996	0.974
	mAP50	0.893	0.696	0.995	0.987
	mAP50-95	0.852	0.684	0.962	0.909
YOLOv10s.pt	Precision (P)	0.997	0.998	1.000	0.992
	Recall (R)	0.884	0.682	0.997	0.974
	mAP50	0.893	0.694	0.995	0.989
	mAP50-95	0.848	0.678	0.961	0.905
YOLOv10m.pt	Precision (P)	0.994	0.996	0.999	0.988
	Recall (R)	0.888	0.682	1.000	0.983
	mAP50	0.892	0.701	0.995	0.981
	mAP50-95	0.852	0.692	0.968	0.897

5.1.1 Summary

Each model demonstrates strengths in different aspects of performance metrics. The YOLOv10s.pt model exhibits the highest precision at 0.997, while the YOLOv10m.pt model shows the highest recall at 0.888. The YOLOv10n.pt and YOLOv10m.pt models have the highest mAP50-95 at 0.852. The selection of an optimal model should consider the specific requirements of precision, recall, and mAP for the intended application.

5.2 Data Description

5.3 Experiment Results

Table 5.2: Performance comparison of various models on trajectory prediction tasks from 30 to 60 frames. The table reports the Average Displacement Error (ADE), Final Displacement Error (FDE), Average Intersection over Union (AIOU), and Final Intersection over Union (FIOU) for each model. Lower ADE and FDE values indicate better accuracy, while higher AIOU and FIOU values indicate better overlap with ground truth. The GRUSizPos model achieves the best performance across all metrics.

Model	**ADE (pxl)**	**FDE (pxl)**	**AIOU (%)**	**FIOU (%)**
CV	131.5	271.0	25.4	8.7
LKF	121.1	270.3	29.3	6.4
LSTMPosVelAcc (ours)	69.1	115.0	26.6	11.2
LSTMSizPos (ours)	49.7	95.7	41.3	15.7
GRUPosVelAcc (ours)	81.5	121.3	23.3	10.7
GRUSizPos (ours)	**32.7**	**73.3**	**47.6**	**20.6**

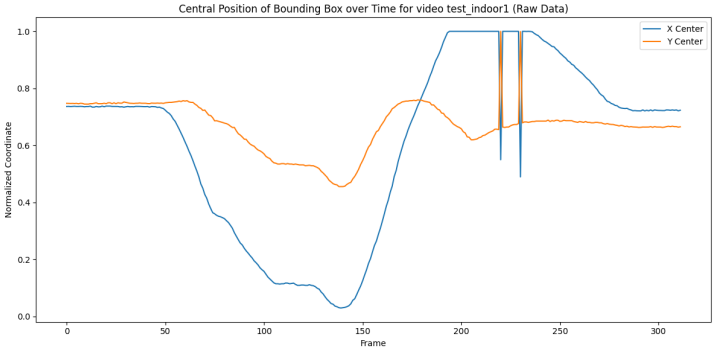
5.4 Testing Visualisation

Chapter 6

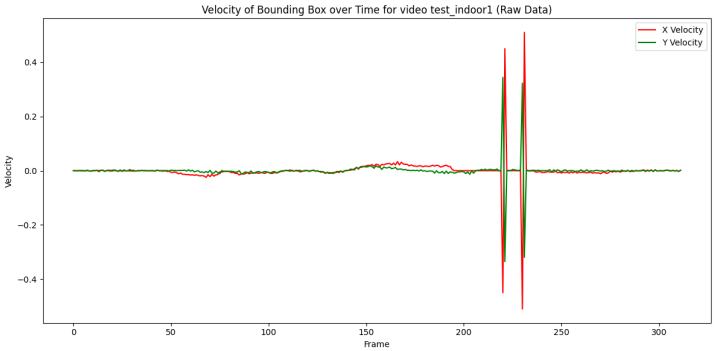
Conclusion and Future Work

Appendix A

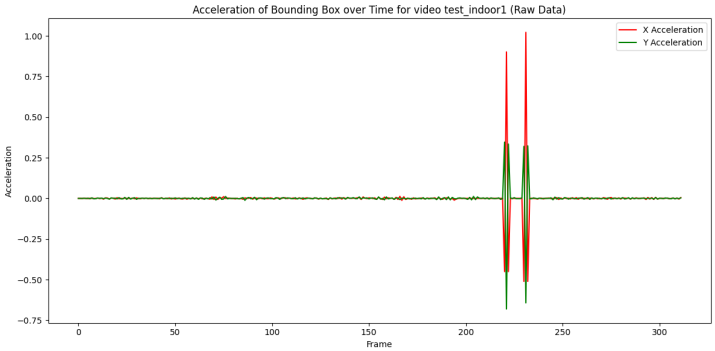
Analysis of Bounding Box Metrics



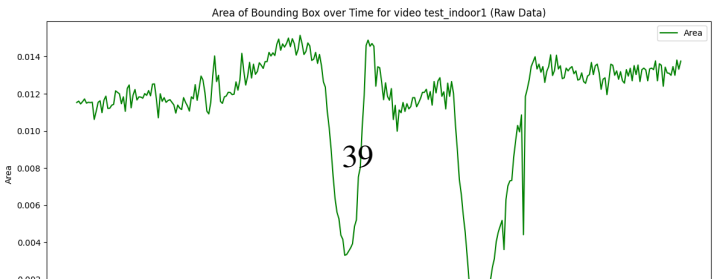
(a) Central position of the refueling port over time.

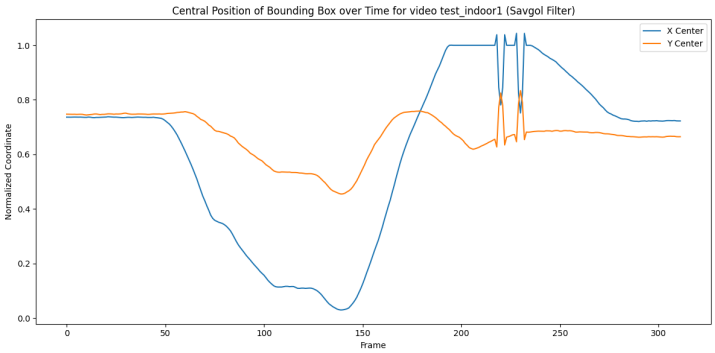


(b) Velocity of the refueling port over time.

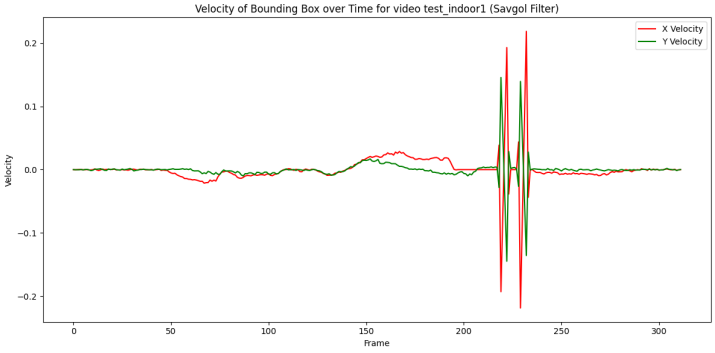


(c) Acceleration of the refueling port over time.

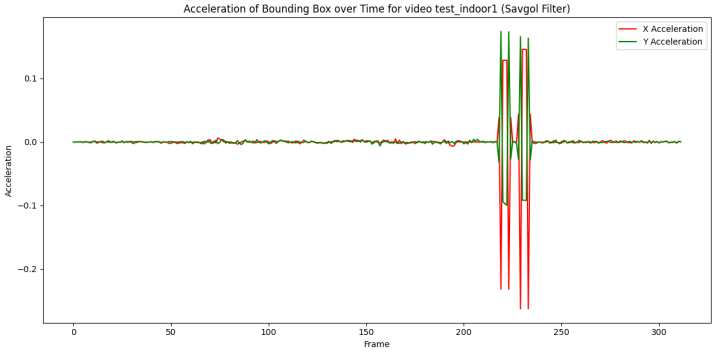




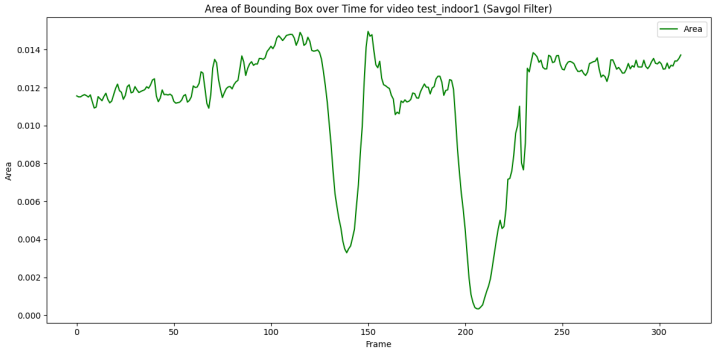
(a) Central position of the refueling port over time.



(b) Velocity of the refueling port over time.

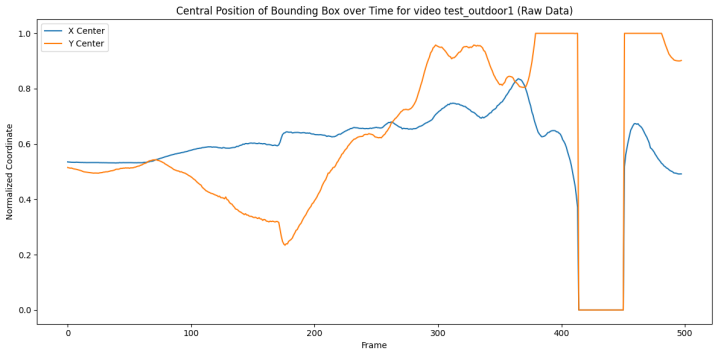


(c) Acceleration of the refueling port over time.

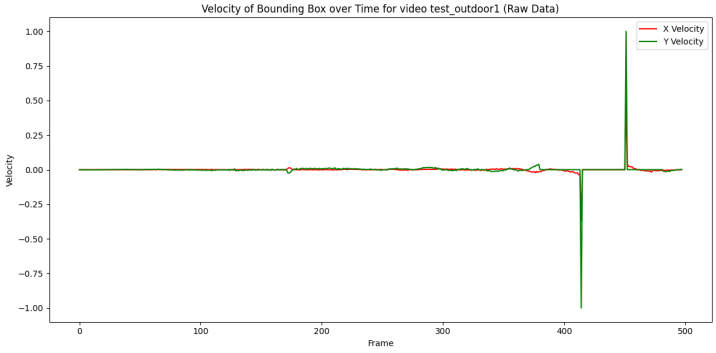


(d) Area of the refueling port over time.

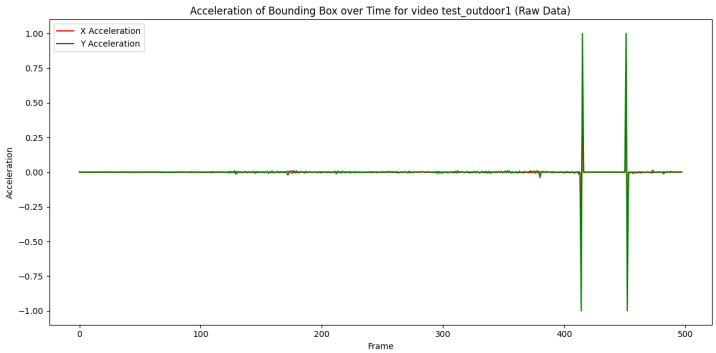
Figure A.2: Temporal analysis of different metrics for the refueling port in the *test_indoor1* video. The metrics include (a) central position, (b) velocity, (c) acceleration, and (d) area, providing a comprehensive overview of the object's dynamics over time.



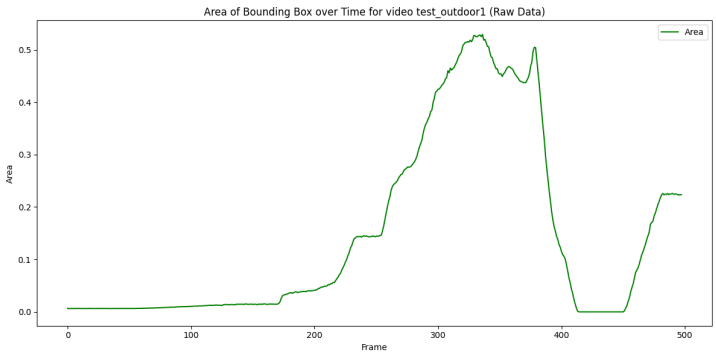
(a) Central position of the refueling port over time.



(b) Velocity of the refueling port over time.

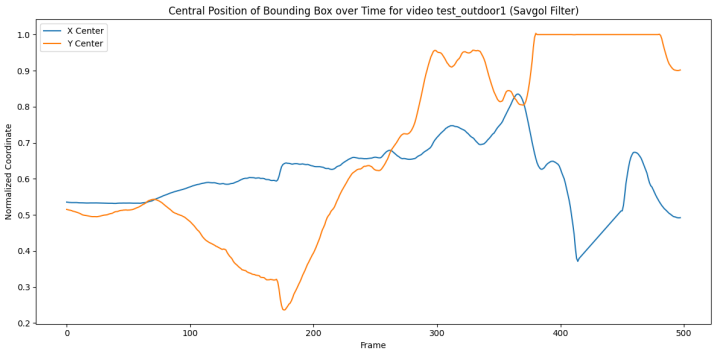


(c) Acceleration of the refueling port over time.

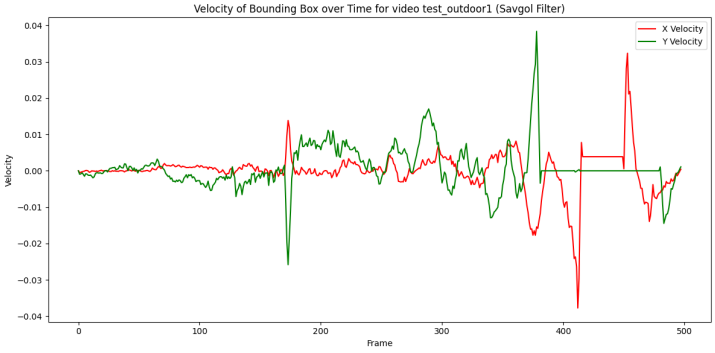


(d) Area of the refueling port over time.

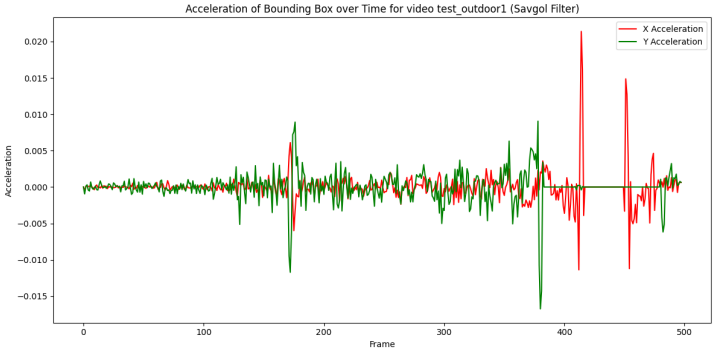
Figure A.3: Temporal analysis of different metrics for the refueling port in the *test_outdoor1* video. The metrics include (a) central position, (b) velocity, (c) acceleration, and (d) area, providing a comprehensive overview of the object's dynamics over time.



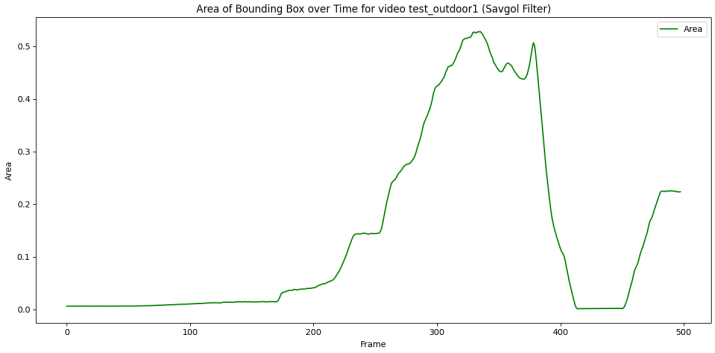
(a) Central position of the refueling port over time.



(b) Velocity of the refueling port over time.

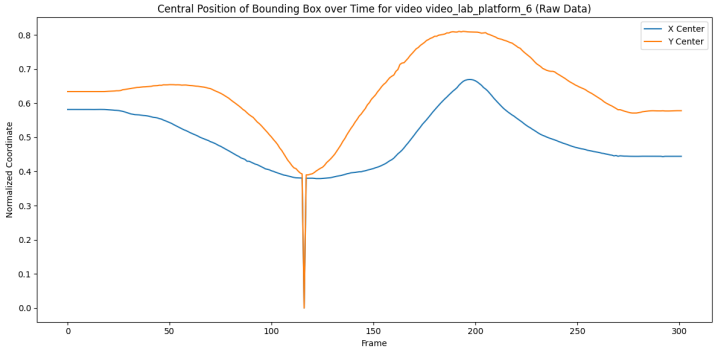


(c) Acceleration of the refueling port over time.

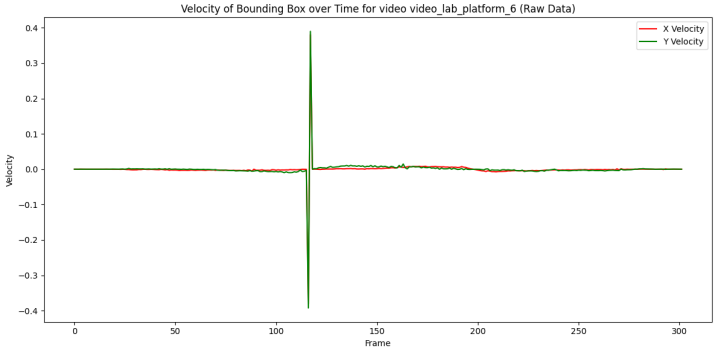


(d) Area of the refueling port over time.

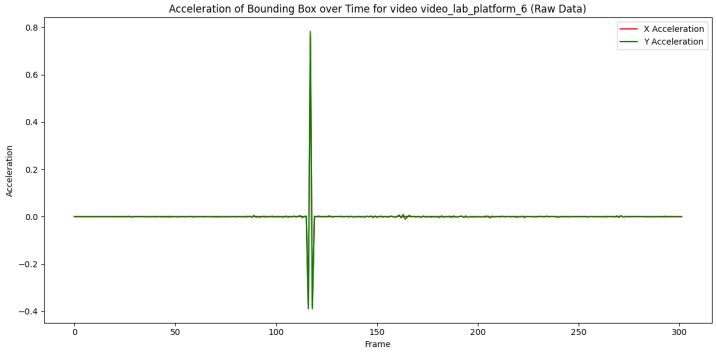
Figure A.4: Temporal analysis of different metrics for the refueling port in the *test_outdoor1* video. The metrics include (a) central position, (b) velocity, (c) acceleration, and (d) area, providing a comprehensive overview of the object's dynamics over time.



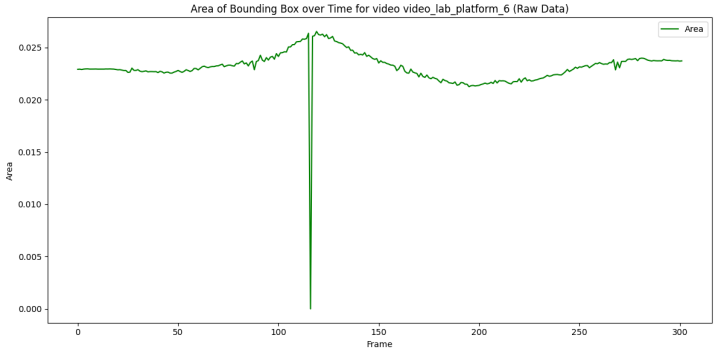
(a) Central position of the refueling port over time.



(b) Velocity of the refueling port over time.

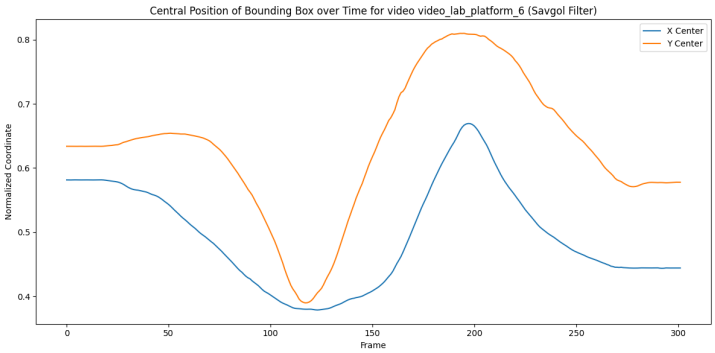


(c) Acceleration of the refueling port over time.

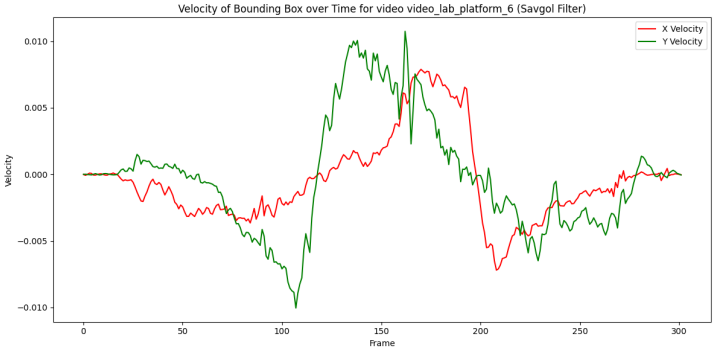


(d) Area of the refueling port over time.

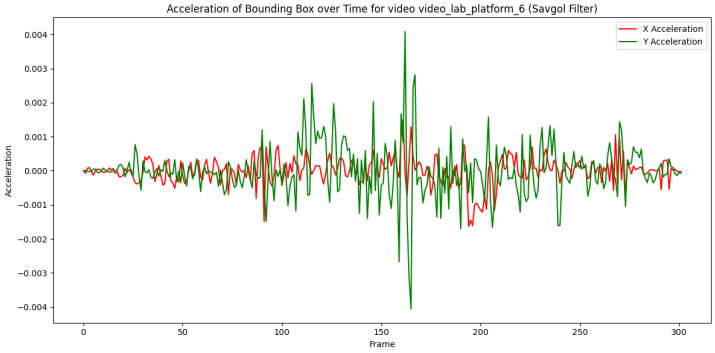
Figure A.5: Temporal analysis of different metrics for the refueling port in the *test_video_lab_platform_6* video. The metrics include (a) central position, (b) velocity, (c) acceleration, and (d) area, providing a comprehensive overview of the object's dynamics over time.



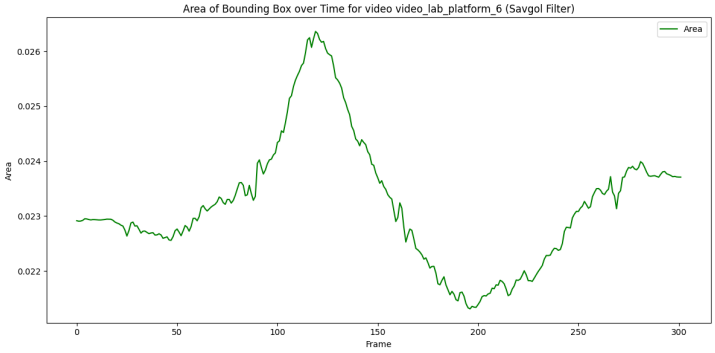
(a) Central position of the refueling port over time.



(b) Velocity of the refueling port over time.



(c) Acceleration of the refueling port over time.



(d) Area of the refueling port over time.

Figure A.6: Temporal analysis of different metrics for the refueling port in the *test_video_lab_platform_6* video. The metrics include (a) central position, (b) velocity, (c) acceleration, and (d) area, providing a comprehensive overview of the object's dynamics over time.

References

1. A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social lstm: Human trajectory prediction in crowded spaces. volume 2016-December, 2016. doi: 10.1109/CVPR.2016.110.

2. A. F. C. A. C. A. . M. C. (AvMC). Ar3p program background (2017-2019) and robotic hot refueling demonstrations (sep-oct 2020), 2020. URL hhttps://www.denix.osd.mil/ndcee/denix-files/sites/44/2023/09/EXSUM-AR3P-Robotic-Refueling-K-MAX-and-S-70-Sep_Oct-2020-v9.pdf. Accessed: 2024-06-24.

3. J. Bailey. What is fuel tankering and why should you care?, 2019. URL https://simpleflying.com/fuel-tankering/. Accessed: 2024-06-24.

4. R. A. Bennett, Y. C. Shiu, and M. B. Leahy. A robust light invariant vision system for aircraft refueling. volume 1, 1991. doi: 10.1109/robot.1991.131568.

5. S. Blakey, L. Rye, and C. W. Wilson. Aviation gas turbine alternative fuels: a review. *Proceedings of the Combustion Institute*, 33(2):2863–2885, 2011. doi: 10.1016/j.proci.2010.09.011.

6. A. Bochkovskiy, C. Wang, and H. M. Liao. Yolov4: Optimal speed and accuracy of object detection. *CoRR*, abs/2004.10934, 2020. URL https://arxiv.org/abs/2004.10934.

7. S. A. Bouhsain, S. Saadatnejad, and A. Alahi. Pedestrian intention prediction: A multi-task perspective. *CoRR*, abs/2010.10270, 2020. URL https://arxiv.org/abs/2010.10270.

8. H. Burnette. Lab demonstrates robotic ground refueling of aircraft, Oct. 2010. URL https://www.wpafb.af.mil/News/Article-Display/Article/400022/lab-demonstrates-robotic-ground-refueling-of-aircraft/. Accessed: 2024-06-24.

9. F. Chen, X. Wang, Y. Zhao, S. Lv, and X. Niu. Visual object tracking: A survey. *Computer Vision and Image Understanding*, 222, 9 2022. ISSN 1090235X. doi: 10.1016/j.cviu.2022.103508.

10. Y. Chen, X. Yuan, R. Wu, J. Wang, Q. Hou, and M.-M. Cheng. Yolo-ms: Rethinking multi-scale representation learning for real-time object detection, 2023.

11. G. Cheng, X. Yuan, X. Yao, K. Yan, Q. Zeng, X. Xie, and J. Han. Towards large-scale small object detection: Survey and benchmarks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45, 2023. ISSN 19393539. doi: 10.1109/TPAMI.2023.3290594.

12. O. Cokorilo, S. Gvozdenovic, L. Vasov, and P. Mirosavljevic. Costs of unsafety in aviation. *Technological and Economic Development of Economy - TECHNOL ECON DEV ECON*, 16:188–201, 06 2010. doi: 10.3846/tede.2010.12.

13. J. Dai, Y. Li, K. He, and J. Sun. R-FCN: object detection via region-based fully convolutional networks. *CoRR*, abs/1605.06409, 2016. URL http://arxiv.org/abs/1605.06409.

14. H. Face. Object detection leaderboard, 2023. URL https://huggingface.co/blog/object-detection-leaderboard. Accessed: 2024-06-17.

15. N. Ficken. Concept demonstration explores robotic aviation refueling system, July 18 2017. URL https://www.army.mil/article/190980/concept_demonstration_explores_robotic_aviation_refueling_system. Accessed: 2024-06-24.

16. Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun. YOLOX: exceeding YOLO series in 2021. *CoRR*, abs/2107.08430, 2021. URL https://arxiv.org/abs/2107.08430.

17. A. Ghosh. Yolov10: The dual-head og of yolo series, 2024. URL https://learnopencv.com/yolov10/. Accessed: 2024-06-24.

18. R. B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015. URL http://arxiv.org/abs/1504.08083.

19. R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013. URL http://arxiv.org/abs/1311.2524.

20. K. Gong, B. Liu, X. Xu, Y. Xu, Y. He, Z. Zhang, and J. Rasol. Research of an unmanned aerial vehicle autonomous aerial refueling docking method based on binocular vision. *Drones*, 7, 2023. ISSN 2504446X. doi: 10.3390/drones7070433.

21. Intel. Intel realsense depth camera d435, 2024. URL https://www.intelrealsense.com/depth-camera-d435/. Accessed: 2024-06-24.

22. G. Jocher. Yolov5 release v7.0, 2022. URL https://github.com/ultralytics/yolov5/tree/v7.0.

23. G. Jocher. Yolov8, 2023. URL https://github.com/ultralytics/ultralytics/tree/main.

24. J. Kang, S. Tariq, H. Oh, and S. Woo. A survey of deep learning-based object detection methods and datasets for overhead imagery. *IEEE Access*, 10:1–1, 01 2022. doi: 10.1109/ACCESS.2022.3149052.

25. M. M. Karim, R. Qin, and Y. Wang. Fusion-gru: A deep learning model for future bounding box prediction of traffic agents in risky driving videos. *Transportation Research Record*, 2024. ISSN 21694052. doi: 10.1177/03611981241230540.

26. B. Kuang, S. Barnes, G. Tang, and K. Jenkins. A dataset for autonomous aircraft refueling on the ground (agr). 2023. doi: 10.1109/ICAC57885.2023.10275212.

27. J. Kugarajeevan, T. Kokul, A. Ramanan, and S. Fernando. Transformers in single object tracking: An experimental survey. *IEEE Access*, 11, 2023. ISSN 21693536. doi: 10.1109/ACCESS.2023.3298440.

28. W. C. Lee, Y. B. Jeon, S. S. Han, and C. S. Jeong. Position prediction in space system for vehicles using artificial intelligence. *Symmetry*, 14, 2022. ISSN 20738994. doi: 10.3390/sym14061151.

29. C. Li, L. Li, Y. Geng, H. Jiang, M. Cheng, B. Zhang, Z. Ke, X. Xu, and X. Chu. Yolov6 v3.0: A full-scale reloading, 2023.

30. T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection, 2018.

31. K. Liu, L. Chen, and X. Liu. Research on application of frontier technologies at smart airport. In J. Zeng, P. Qin, W. Jing, X. Song, and Z. Lu, editors, *Data Science*, pages 319–330, Singapore, 2021. Springer Singapore. ISBN 978-981-16-5943-0.

32. S. Liu, D. Liu, G. Srivastava, D. Połap, and M. Woźniak. Overview and methods of correlation filter algorithms in object tracking. *Complex and Intelligent Systems*, 7, 2021. ISSN 21986053. doi: 10.1007/s40747-020-00161-4.

33. W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015. URL http://arxiv.org/abs/1512.02325.

34. M. D. L. Olja Čokorilo and G. Dell'Acqua. Aircraft safety analysis using clustering algorithms. *Journal of Risk Research*, 17(10):1325–1340, 2014. doi: 10.1080/13669877.2013.879493. URL https://doi.org/10.1080/13669877.2013.879493.

35. E. Plaza and M. Santos. Knowledge based approach to ground refuelling optimization of commercial airplanes. *Expert Systems*, 38, 2021. ISSN 14680394. doi: 10.1111/exsy.12631.

36. J. Redmon and A. Farhadi. YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242, 2016. URL http://arxiv.org/abs/1612.08242.

37. J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015. URL http://arxiv.org/abs/1506.02640.

38. S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39, 2017. ISSN 01628828. doi: 10.1109/TPAMI.2016.2577031.

39. R. Sati, S. Singh, and R. Yadav. Aircraft fuel system: design, components, and safety. *International Journal of Aerospace Engineering*, 2019:1–12, 2019. doi: 10.1155/2019/6943787.

40. E. R. Schultz. Robotic systems for aircraft servicing/maintenance. *IEEE Aerospace and Electronic Systems Magazine*, 1, 1986. ISSN 08858985. doi: 10.1109/MAES.1986.5005018.

41. O. Styles, T. Guha, and V. Sanchez. Multiple object forecasting: Predicting future object locations in diverse environments. 2020. doi: 10.1109/WACV45572.2020.9093446.

42. A. Wang, H. Chen, L. Liu, K. Chen, Z. Lin, J. Han, and G. Ding. Yolov10: Real-time end-to-end object detection, 2024.

43. C. Wang, W. He, Y. Nie, J. Guo, C. Liu, K. Han, and Y. Wang. Gold-yolo: Efficient object detector via gather-and-distribute mechanism, 2023.

44. C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao. Yolov9: Learning what you want to learn using programmable gradient information, 2024.

45. X. Wang, X. Dong, X. Kong, J. Li, and B. Zhang. Drogue detection for autonomous aerial refueling based on convolutional neural networks. *Chinese Journal of Aeronautics*, 30, 2017. ISSN 10009361. doi: 10.1016/j.cja.2016.12.022.

46. J. Wu and S. Xu. From point to region: Accurate and efficient hierarchical small object detection in low-resolution remote sensing images. *Remote Sensing*, 13, 2021. ISSN 20724292. doi: 10.3390/rs13132620.

47. S. Xu, X. Wang, W. Lv, Q. Chang, C. Cui, K. Deng, G. Wang, Q. Dang, S. Wei, Y. Du, and B. Lai. Pp-yoloe: An evolved version of yolo, 2022.

48. X. Xu, Y. Jiang, W. Chen, Y. Huang, Y. Zhang, and X. Sun. Damo-yolo : A report on real-time object detection design, 2023.

49. T. Ye, W. Qin, Z. Zhao, X. Gao, X. Deng, and Y. Ouyang. Real-time object detection network in uav-vision based on cnn and transformer. *IEEE Transactions on Instrumentation and Measurement*, 72, 2023. ISSN 15579662. doi: 10.1109/TIM.2023.3241825.

50. S. Yildirim, Z. Rana, and G. Tang. Autonomous ground refuelling approach for civil aircrafts using computer vision and robotics. volume 2021-October, 2021. doi: 10.1109/DASC52595.2021.9594312.

51. S. Yildirim, Z. A. Rana, and G. Tang. Development of vision guided real-time trajectory planning system for autonomous ground refuelling operations using hybrid dataset. 2023. doi: 10.2514/6.2023-1148.

52. S. S. A. Zaidi, M. S. Ansari, A. Aslam, N. Kanwal, M. Asghar, and B. Lee. A survey of modern deep learning based object detection models, 2022. ISSN 10512004.

53. Y. Zhang, T. Wang, K. Liu, B. Zhang, and L. Chen. Recent advances of single-object tracking methods: A brief survey. *Neurocomputing*, 455, 2021. ISSN 18728286. doi: 10.1016/j.neucom.2021.05.011.

54. Z. Zhong, D. Li, H. Wang, and Z. Su. Drogue position and tracking with machine vision for autonomous air refueling based on ekf. volume 2, 2017. doi: 10.1109/IHMSC.2017.151.